DETECTION AND TRACKING OF STEALTHY TARGETS USING PARTICLE FILTERS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

by

Philip Losie

December 2009

COMMITTEE MEMBERSHIP

TITLE: Detection and Tracking of Stealthy Targets using Particle Filters

AUTHOR: Philip Losie

DATE SUBMITTED: December 2009

COMMITTEE CHAIR:	Dr. John Saghri, Associate Professor
COMMITTEE MEMBER:	Dr. Helen Yu, Associate Professor
COMMITTEE MEMBER:	Dr. Jane Zhang, Associate Professor

ABSTRACT

Detection and Tracking of Stealthy Targets using Particle Filters

Philip Losie

In recent years, the particle filter has gained prominence in the area of target tracking because it is robust to non-linear target motion and non-Gaussian additive noise. Traditional track filters, such as the Kalman filter, have been well studied for linear tracking applications, but perform poorly for non-linear applications. The particle filter has been shown to perform well in non-linear applications. The particle filter method is computationally intensive and advances in processor speed and computational power have allowed this method to be implemented in real-time tracking applications.

This thesis explores the use of particle filters to detect and track stealthy targets in noisy imagery. Simulated point targets are applied to noisy image data to create an image sequence. A particle filter method known as Track-Before-Detect is developed and used to provide detection and position tracking estimates of a single target as it moves in the image sequence. This method is then extended to track multiple moving targets. The method is analyzed to determine its performance for targets of varying signal-to-noise ratio and for varying particle set sizes.

The simulation results show that the Track-Before-Detect method offers a reliable solution for tracking stealthy targets in noisy imagery. The analysis shows that the proper selection of particle set size and algorithm improvements will yield a filter that can track targets in low signal-to-noise environments. The multi-target simulation results show that the method can be extended successfully to multi-target tracking applications.

This thesis is a continuation of automatic target recognition and target tracking research at Cal Poly under Dr. John Saghri and is sponsored by Raytheon Space and Airborne Systems.

Keywords: particle filter, target tracking, track-before-detect, Kalman filter

ACKNOWLEDGEMENTS

I would like to express sincere gratitude to my thesis advisor, Dr. John Saghri, for his guidance, patience, and support. I thank him for selecting me to be a part of his research team and for the opportunity to continue research into the field of target tracking.

I would also like to thank Raytheon Space and Airborne Systems for their sponsorship and encouragement of this work.

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction

1.1. Target Tracking System

The objective of target tracking is to continuously and accurately estimate a target's current position, velocity, and acceleration given a measurement. The target tracking system consists of a sensor, signal processing, data processing, and control [1].

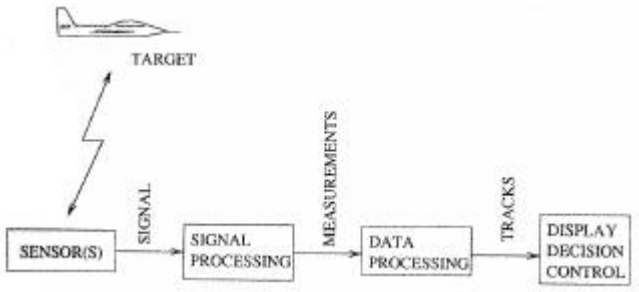

Figure 1-1: Target tracking system

The sensors take in observations of the target and pass them to a signal processing component. The signal processing component produces a measurement. The data processing component uses the measurement to construct a track of the target. Tracks are a sequence of target state estimates. The target state may be position, velocity, or acceleration. The control component converts the target tracks to display to the user and applies decision logic.

1.2. Sensors

A wide range of sensors are used in many target tracking applications. Typical sensors used in target tracking for military applications range from airborne tactical air-to-air radar, synthetic aperture radar (SAR), ground moving target indicator (GMTI) radar, and forward looking infrared (FLIR) cameras.

1.3. Tracking Filters

This thesis focuses on the data processing component that uses one or more track filters to recursively estimate the target state. Commonly used track filters are alpha-beta filters, H infinity filters, Kalman filters, and recently particle filters [2]. Track filters play a crucial role in target tracking system performance. They determine how well the system can estimate the true target state given noisy measurements. Alpha-beta and H infinity filters are discussed in Sections 1.4 and 1.5. The Kalman and particle filters are discussed in detail in Sections 2 and 3.

1.4. Alpha-beta Filters

The alpha-beta filter is a much simplified version of the Kalman filter. It assumes that the target dynamics can be modeled by a two-state system. It is defined by the following equations:

$$x_k^s = x_k^p + \alpha\left(x_k^m - x_k^p\right)$$
$$v_k^s = v_{k-1}^s + \frac{\beta\left(x_k^m - x_k^p\right)}{\Delta T}$$
$$x_{k+1}^p = \frac{x_k^p}{v_k^s \Delta T}$$

where at time *k* the smoothed position estimate is x_k^s, the predicted position is x_k^p, the measured position estimate is x_k^m, and the smoothed velocity estimate is v_k^s. ΔT is the time interval between measurements. The filter parameters, α and β, are typically chosen to be between 0 and 1. Suggested starting values for these parameters are $0.1 < \alpha < 0.2$ and $0.05 < \beta < 0.1$ [3]. The parameter β should be small so that the velocity estimate is not excessively sensitive to error in position measurements. A weakness of this filter is that the filter parameters are determined experimentally, unlike the Kalman filter which computes the Kalman gain recursively at each time step. Therefore, the filter is sensitive to variations in measurement noise.

1.5. H Infinity Filters

The H infinity filter is a more robust version of the Kalman filter [3]. The Kalman filter minimizes the variance of the estimation error and assumes known noise properties. The H infinity filter makes no assumptions about the noise present in the system and minimizes the worst case estimation error. Many different variations of the H infinity filter exist.

Given a discrete linear system defined as:

$$x_{k+1} = Ax_k + Bu_k + w_k$$
$$y_k = Cx_k + z_k$$

where the matrices *A*, *B*, and *C* are known, *k* is the time step, *x* is the system state, *u* is the input, *y* is the measured output, and *w* and *z* are system noise. The H infinity filter seeks to find a state estimate that minimizes the worst possible effect of *w* and *z* on the system. This filter is often called the *minimax* filter because it seeks to minimize the maximum estimation error.

The filter equations are as follows:

$$
\begin{aligned}
L_k &= \left(I - QP_k + C^T V^{-1} C P_k\right)^{-1} \\
K_k &= A P_k L_k C^T V^{-1} \\
\hat{x}_{k+1} &= A\hat{x}_k + Bu_k + K_k(y_k - C\hat{x}_k) \\
P_{k+1} &= A P_k L_k A^T + W
\end{aligned}
$$

where $\hat{x}_k$ is the state estimate at time step *k*, P_k is the estimation error covariance, and K_k is the H infinity gain. The initial state estimate should be set to the designer's best guess and the initial error covariance should be set for appropriate filter performance. This filter can be difficult to implement due to the fact that there are numerous parameters that need to be tuned to achieve an acceptable performance level.

1.6. Target Tracking at Cal Poly

Previous work by Cal Poly students, under the guidance of Dr. John Saghri and sponsorship of Raytheon Space and Airborne Systems, has focused on the areas of automatic target recognition (ATR) and tracking of targets in SAR imagery. Jessica Kiefer's thesis explored the application of various track filters to moving targets in SAR imagery [3]. In her work, Kalman, H Infinity, and Prediction and Matching Detection (PAMD) filters were analyzed for their tracking

performance. This thesis will focus on the use of particle filters in tracking moving targets.

2. Kalman Filtering

The Kalman filter is the most widely used estimation approach in a variety of fields. It is a very common technique for target tracking in the field of radar. The Kalman filter is a linear estimator that provides a minimized mean square error solution to general linear estimation problems. It is simple to implement and computationally efficient. For the case of nonlinear systems, the extended Kalman filter (EKF) is a common solution where the nonlinear system is linearized about the current system state.

2.1. Kalman Filter Algorithm

Given a general linear state space system:

$$x_k = F_k x_{x-1} + w_k$$

where x_k is the system state, F_k is the state transition matrix, and w_{k-1} is the zero-mean white Gaussian process noise with covariance matrix Q_k. At time k we have a measurement according to:

$$z_k = H_k x_k + v_k$$

where z_k is the measurement, H_k is the observation matrix, and v_k is the zero-mean white Gaussian observation noise with covariance matrix R_k.

The Kalman filter algorithm consists of two steps; predict and update. The predict phase uses the state estimate from the previous time step to estimate the current state. The update phase takes in the current measurement and refines the prediction by adjusting a gain factor, known as the Kalman gain. The predict and update phase equations are as follows:

Predict

State estimate: $\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1}$

Covariance estimate: $\hat{P}_{k|k-1} = F_k \hat{P}_{k-1|k-1} F_k^T + Q_{k-1}$

Update

Innovation residual: $y_k = z_k - H_k \hat{x}_{k|k-1}$

Innovation covariance: $S_k = H_k \hat{P}_{k|k-1} H_k^T + R_k$

Kalman gain: $K_k = \hat{P}_{k|k-1} H_k^T S_k^{-1}$

Updated state estimate: $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k$

Updated covariance estimate: $\hat{P}_{k|k} = (I - K_k H_k) \hat{P}_{k|k-1}$

2.2. Extended Kalman Filter Algorithm

Given a general nonlinear version of the state space system above where *f* and *h* are differentiable functions:

$$x_k = f(x_{k-1}) + w_k$$
$$z_k = h(x_k) + v_k$$

where the noise terms w_k and v_k are the same as above. The extended Kalman filter equations are as follows:

Predict

State estimate: $\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1})$

Covariance estimate: $\hat{P}_{k|k-1} = F_k \hat{P}_{k-1|k-1} F_k^T + Q_{k-1}$

Update

Innovation residual: $y_k = z_k - h(\hat{x}_{k|k-1})$

Innovation covariance: $S_k = H_k \hat{P}_{k|k-1} H_k^T + R_k$

Kalman gain: $K_k = \hat{P}_{k|k-1} H_k^T S_k^{-1}$

Updated state estimate: $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k$

Updated covariance estimate: $\hat{P}_{k|k} = (I - K_k H_k) \hat{P}_{k|k-1}$

The matrices F_k and H_k are the Jacobians:

$$F_k = \left[\frac{\partial f}{\partial x} \right]_{x=\hat{x}_{k-1|k-1}}$$

$$H_k = \left[\frac{\partial h}{\partial x} \right]_{x=\hat{x}_{k|k-1}}$$

At each time step, the state transition and observation matrices are linearized about the current state estimate. Though this method provides an approximate solution, it is a very popular approach to nonlinear state estimation.

3. Particle Filter

The traditional approach to state estimation involves the use of the Kalman filter. This approach works well for problems involving a linear system model with Gaussian noise. The general approach is to estimate the state probability density function (PDF) and at every iteration of the Kalman filter, propagate and update the mean and covariance of the estimated distribution. For a linear and Gaussian problem, the PDF remains Gaussian at every iteration. However, for a nonlinear or non-Gaussian problem, there is no analytic expression for the PDF [4]. The extended Kalman filter (EKF) only approximates the PDF with a Gaussian. Particle filtering offers a better solution to nonlinear non-Gaussian problems.

3.1. Bayesian State Estimation

In the Bayesian approach to state estimation, one attempts to estimate the posterior probability density function of the state based on all available information, including past measurements. This PDF encapsulates all statistical information of the system and may be regarded as the complete solution to the estimation problem [5]. In order to understand the Bayesian approach, we must define the state space estimation problem as outlined in [4].

The state vector at time k is defined as x_k and evolves according to the discrete-time stochastic model:

$$x_k = f_{k-1}(x_{k-1}, w_{k-1})$$

where f_{k-1} is a known function of the state vector x_{k-1} and process noise vector w_{k-1}. Measurements of the state, defined as y_k, are related to the state through the measurement equation:

$$y_k = h_k(x_k, v_k)$$

where h_k is a known function of the state vector x_k and the measurement noise vector v_k. We assume that the noise sequences w_{k-1} and v_k are zero mean, white noise with known probability density functions and are independent. It is assumed that the initial PDF of the state vector is known and defined as $p(x_0)$.

The goal is to compute filtered estimates of x_k based upon the sequence of available measurements $Y_k = \{y_i, i = 1,...,k\}$. We can compute these filtered estimates by first obtaining the posterior PDF $p(x_k/Y_k)$. Expressions for the posterior PDF are derived in [4] and are as follows using the Chapman-Kolmogorov equation:

$$p(x_k \mid Y_{k-1}) = \int p(x_k \mid x_{k-1}) p(x_{k-1} \mid Y_{k-1}) dx_{k-1} \quad (1)$$

$$p(x_k \mid Y_k) = \frac{p(y_k \mid x_k) p(x_k \mid Y_{k-1})}{p(y_k \mid Y_{k-1})} \quad (2)$$

where the normalizing constant is defined as

$$p(y_k \mid Y_{k-1}) = \int p(y_k \mid x_k) p(x_k \mid Y_{k-1}) dx_k \quad (3)$$

In eq. (1), $p(x_k/x_{k-1})$ is called the transitional density and is defined by the system model and process noise sequence. In equation (2), $p(y_k/x_k)$ is called the likelihood function and is defined by the measurement equation and measurement noise sequence.

Eqs. (1) and (2) represent the prediction and update stages of the Bayesian approach to state estimation and form the basis for the optimal Bayesian solution [5]. Suppose that $p(x_{k-1}/Y_{k-1})$ is available at time step k-1. Eq. (1) predicts the prior PDF $p(x_k/Y_{k-1})$. Eq. (2) then updates the predicted prior to the current time step k and obtains the posterior PDF $p(x_k/Y_k)$. From the posterior PDF, one can compute optimal state estimates with respect to any criterion [2]. For example, a common state estimate is the minimum mean-square error (MMSE) estimate of the mean x_k:

$$x_k^{MMSE} = \int x_k \, p(x_k \mid Y_k) dx_k$$

The prediction and update stages defined in (1) and (2) of the Bayesian approach is only a conceptual solution to the state estimation problem. In general, analytical expressions for the equations do not exist. The particle filter described in the following sections produces a suboptimal solution to the problem by approximating the prediction and update equations. This solution however is computationally feasible and is robust to most state estimation problems.

3.2. Sampling Importance Resampling Filter

Particle filtering, also known as Sequential Monte Carlo methods, is a recursive Bayesian algorithm for state estimation. In the state space, the posterior PDF $p(x_k/Y_k)$ is represented by a set of random samples instead of an analytic expression. As the number of samples grows, they better approximate the exact PDF. From the samples, estimates of the mean and covariance can be obtained. The most common version of the particle filter is the Sampling Importance Resampling (SIR) filter.

The steps of the SIR filter algorithm are as follows:

Assumptions:

Given a state space system model and measurement equation:

$$x_{k+1} = f_k(x_k, w_k)$$
$$y_k = h_k(x_k, v_k)$$

where w_k and v_k are zero mean, white-noise system and measurement noise with probability density functions, x_k is the state, and y_k is the measurement at time step k. It is assumed that the initial state PDF is known, $p(x_0)$, and the functions f_k and h_k.

Filter Initialization:

A set of N random samples, $\{x_{k-1}(i) : i = 1, \ldots, N\}$ are selected from the PDF $p(x_0)$.

Algorithm:

Perform the following steps 1-3 iteratively for each time step k:

Step 1 - Prediction:

Pass each sample through the system model to obtain samples of the PDF at time k:

$$x_k^*(i) = f_{k-1}(x_{k-1}(i), w_{k-1}(i))$$

where $w_{k-1}(i)$ is a sample drawn from the known system noise PDF.

Step 2 - Update:

Compute the normalized likelihood weight for each sample:

$$q_i = \frac{p(y_k \mid x_k^*(i))}{\sum_{j=1}^{N} p(y_k \mid x_k^*(j))}$$

where $p(y_k \mid x_k^*(i))$ is the likelihood.

Step 3 – Resampling and Estimation:

The set of samples $\{x_k^*(i) : i = 1,..., N\}$ with likelihood weights $\{q_i : i = 1,..., N\}$ represent a discrete probability distribution. Draw N samples from this distribution to create a set of N new samples, $\{x_k(i) : i = 1, \ldots, N\}$ that describe the posterior PDF:

$$p(x_k \mid Y_k) \approx \sum_{i=1}^{N} q_i \delta(x_k - x_k(i))$$

The new samples are independent and identically distributed with uniform weight. Statistics of the posterior PDF can then be computed such as the mean and covariance of the state estimate:

$$\hat{x}_k^{MEAN} = \frac{1}{N} \sum_{i=1}^{N} x_k(i)$$

$$\hat{\sigma}_k^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_k(i) - x_k^{MEAN})(x_k(i) - x_k^{MEAN})^T$$

Two iterations of the SIR algorithm can be visualized in Figure 3-1 for a set of 10 particles.

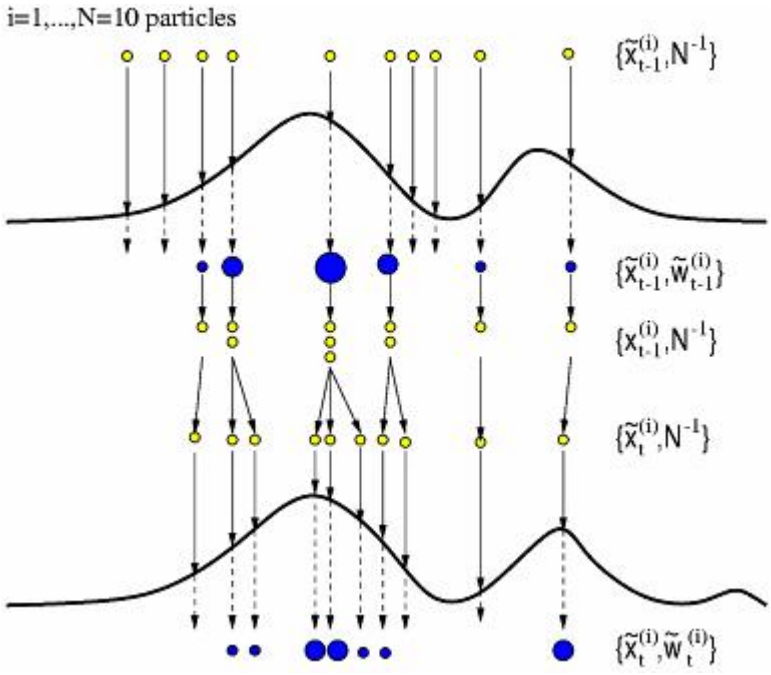

Figure 3-1: SIR algorithm steps

At time *t-1*, the particle set $\tilde{x}_{t-1}^{(i)}$ with equal weights N^{-1} is passed through the system model (Step 1). The yellow dots represent the particles. The weights $\tilde{w}_{t-1}^{(i)}$ are then calculated (Step 2), represented by the blue dots. Large weights correspond to large blue dots in the figure. The particle set is then resampled (Step 3) yielding a new set $x_{t-1}^{(i)}$ of equal weight N^{-1}.

In the SIR algorithm described above, there is a tradeoff associated with the choice of N. A large N will produce a better approximation to the posterior PDF, but at the cost of increased computational load.

3.3. Resampling

The main idea in the resampling step in the SIR algorithm is to eliminate particles with small weights and multiply particles with large weights. This step plays a critical role in the performance of the filter. Without this step, the filter

would suffer from a problem known as sample degeneracy [5]. After a few iterations of the SIR filter without resampling, all but one of the particles will have near-zero weight. This is due to the fact that the variance of the weights can only increase over time [5]. The particles with negligible weight offer no contribution to the estimation of the posterior PDF. Degeneracy of a set of particles can be quantified by the effective sample size:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} q_i^2}$$

Where q_i is the normalized weight and $1 \le N_{eff} \le N$. For the case of uniform particle weights, $N_{eff} = N$. For the extreme degeneracy case of one particle with weight equal to one and all other particles with weight zero, $N_{eff} = 1$. Some versions of the particle filter test N_{eff} against a chosen threshold value to determine if resampling is needed on each iteration of the filter. The generic SIR algorithm above resamples on every iteration.

The resampling algorithm is as follows:

1. **For** $i = 1,\ldots, N$:
   - Draw a random sample u_i from the uniform distribution interval $U[0,1]$.
   - Find the value M that satisfies eq. (4) is true and $q_0 = 0$.

$$\sum_{j=0}^{M-1} q_j < u_i \le \sum_{j=0}^{M} q_j \qquad (4)$$

Select the new sample value: $x_k(i) = x_k^*(M)$

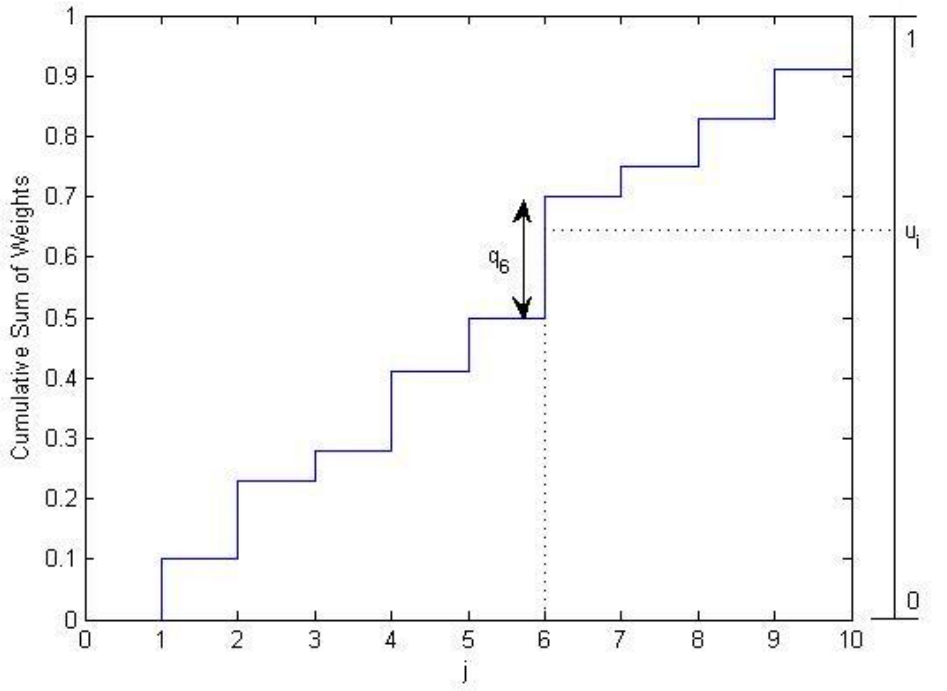

Figure 3-2: Example of resampling for ten particles

Eq. (4) can be represented graphically in Figure 3-2. For each particle, in this case there are 10 total, a random number u_i is chosen and matched to the corresponding weight q_j. The new sample is selected as $x_k^*(6)$. Particles with larger weights will be chosen more often than particles with smaller weights.

The resampling step requires significant time to resample all of the particles if the total particle count N is very large. The algorithm described above has a complexity of *O*(N log N) and is not very efficient. Much research has been performed to develop faster resampling algorithms for real-time applications [6]. Systematic Resampling (SR) is very common in practical applications of particle filters and offers two advantages: it is simple to implement and has *O*(N) complexity. The details of the SR algorithm are as follows [4]:

1. Construct the cumulative sum of weights for $i = 1,\ldots,N$:

- $c_i = c_{i-1} + q_i$ where $c_1 = q_1$

2. Select an initial value *u* from the uniform distribution *U*[0, N^{-1}].
3. **For** *j = 1,…, N* :
   - Increment the initial value: $u_j = u + \frac{j-1}{N}$
   - **While** $u_j > c_i$:
     - *i = i + 1*
   - Selected new sample: $x_k(j) = x_k^*(i)$

3.4. Regularized Particle Filter

The SIR algorithm possesses a major flaw in the diversity of its particles as the filter iterates over time. After resampling, particles with large weights will be selected and multiplied many times. Particles with small weights will eventually disappear. This leads to a loss of diversity among the particle set and is extreme in the case of small process noise [5]. There are many techniques to counteract this problem. The most common method is to add a *regularization* step to the SIR algorithm, known as the Regularized Particle Filter (RPF).

The RPF differs from the SIR algorithm only in the resampling step of the filter. The RPF adds a calculated value to the resampled set:

$$x_k(i) = RESAMPLED\{x_k^*(i)\} + h_{opt} D_k \varepsilon^i \text{ for } i = 1,\ldots, N$$

where D_k is related to the covariance matrix of the particle set, h_{opt} is computed from D_k, and ε^i is computed from a Gaussian kernel [7].

This step jitters the resampled set of particles, increasing their diversity. The jittered set, however, is no longer a true representation of the posterior PDF. The parameters ε^i and h_{opt} are chosen such that they minimize the mean integrated square error (MISE) between the true posterior PDF and the jittered estimate [7].

4. Modeling of Dynamic Target Motion

Tracking algorithm performance depends in part upon the chosen models for target motion. Capturing all possible real-world target motions is a nontrivial task. Two of the most common target models are the constant velocity and constant acceleration kinematic models. The coordinated turn model is useful for highly maneuverable targets such as aircraft. In these models, the process noise covariance matrix *Q* represents uncertainty in state estimation due to random target motion.

4.1. Constant Velocity Model

The constant velocity (CV) model uses white noise for the target acceleration. The state equation for the model is as follows with *T* being the sampling period [8]:

$$x_k = Fx_{k-1} + \Gamma w_{k-1}$$

$$\text{where } x_k = \begin{bmatrix} x \\ \dot{x} \end{bmatrix},\ w_k = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix},\ F = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix},\ \Gamma = \begin{bmatrix} 0.5T^2 \\ T \end{bmatrix}$$

The covariance of the process noise w_{k-1} is defined as:

$$Q = E\left[\Gamma w_{k-1} w_{k-1} \Gamma^T\right] = \Gamma \sigma_w^2 \Gamma = \begin{bmatrix} 0.25T^4 & 0.5T^3 \\ 0.5T^3 & T^2 \end{bmatrix} \sigma_w^2$$

4.2. Constant Acceleration Model

The constant acceleration (CA) model is similar to the CA model with an extension to acceleration [8]:

$$x_k = Fx_{k-1} + \Gamma w_{k-1}$$

$$\text{where } x_k = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix},\ w_k = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix},\ F = \begin{bmatrix} 1 & T & 0.5T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix},\ \Gamma = \begin{bmatrix} 0.5T^2 \\ T \\ 1 \end{bmatrix}$$

The covariance of the process noise is now:

$$Q = \Gamma \sigma_w^2 \Gamma = \begin{bmatrix} 0.25T^4 & 0.5T^3 & 0.5T^2 \\ 0.5T^3 & T^2 & T \\ 0.5T^2 & T & 1 \end{bmatrix} \sigma_w^2$$

4.3. Coordinated Turn Model

The coordinated turn model can be useful to represent a maneuvering target. This model describes a target performing a circular turn in two dimensions. The position and velocity state equations are as follows assuming a turning target in the x-y plane [9]:

$$\begin{aligned} x_{k+1} &= x_k + T\left[SW\dot{x}_k - CW\dot{y}_k\right] \\ y_{k+1} &= y_k + T\left[CW\dot{x}_k + SW\dot{y}_k\right] \\ \dot{x}_{k+1} &= \dot{x}_k \cos(\omega T) - \dot{y}_k \sin(\omega T) \\ \dot{y}_{k+1} &= \dot{x}_k \sin(\omega T) + \dot{y}_k \cos(\omega T) \end{aligned}$$

where ω is the turn rate. SW and CW are defined as:

$$SW = \frac{\sin(\omega T)}{\omega T}$$

$$CW = \frac{1 - \cos(\omega T)}{\omega T}$$

If we define the full state as $x_k = [x \quad \dot{x} \quad y \quad \dot{y} \quad \omega]^T$ then we can define the complete model as[2]:

$$F = \begin{bmatrix} 1 & \frac{\sin(\omega T)}{\omega} & 0 & \frac{-(1-\cos(\omega T))}{T} & \frac{\partial x}{\partial \omega} \\ 0 & \cos(\omega T) & 0 & -\sin(\omega T) & \frac{\partial \dot{x}}{\partial \omega} \\ 0 & \frac{(1-\cos \omega T)}{\omega} & 1 & \frac{\sin(\omega T)}{\omega} & \frac{\partial y}{\partial \omega} \\ 0 & \sin(\omega T) & 0 & \cos(\omega T) & \frac{\partial \dot{y}}{\partial \omega} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q = q \begin{bmatrix} B & 0_{2X2} & 0_{2X1} \\ 0_{2X2} & B & 0_{2X1} \\ 0_{1X2} & 0_{1X2} & \frac{\sigma_\omega^2 T^2}{q} \end{bmatrix} \text{ where } B = \begin{bmatrix} 0.25T^4 & 0.5T^3 \\ 0.5T^3 & T^2 \end{bmatrix} \text{ and } 0_{2X2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

4.4. Simulation Example

Figure 4-1 shows a comparison of the three target motion models in a 10 sample period simulation. All targets started with the same initial x and y velocity and zero process noise. The CA model target was given accelerations a_x and a_y of 0.5 and the turning model target was given a turn rate of 0.1.

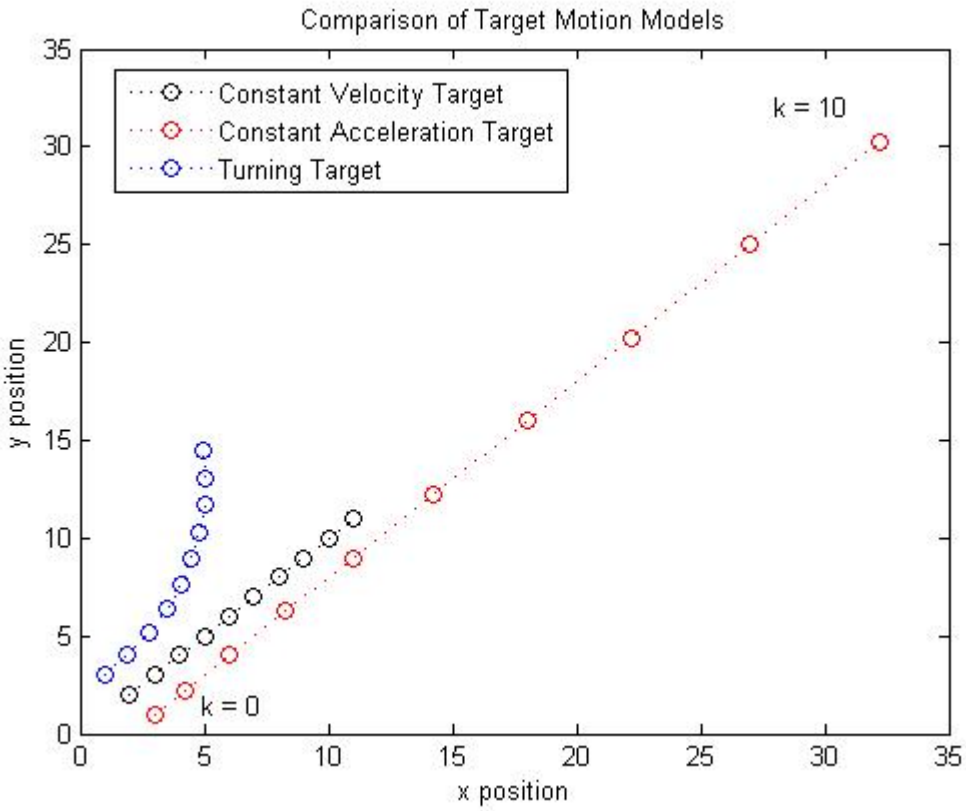

Figure 4-1: Target motion model simulation

5. Target Tracking Scenarios

The following scenario examples illustrate the performance of the Kalman and particle filters for linear and nonlinear systems.

5.1. Linear Scenario

In this scenario, a target moves with constant velocity in the x-y plane. Noisy position observations are taken at each time step *k*. The scenario runs for 20 time steps. The target motion is modeled in the state transition matrix using the constant velocity model with white Gaussian additive noise. The measurements are linear with white Gaussian additive noise also. The initial state vector is:

$$x_0 = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 10 \\ 1 \\ 10 \\ 0 \end{bmatrix}$$

with process noise $\sigma_w^2 = 1$ and observation noise $\sigma_v^2 = 1$. Figure 5-1 shows a realization of the true target position at each time step and the associated position observations.

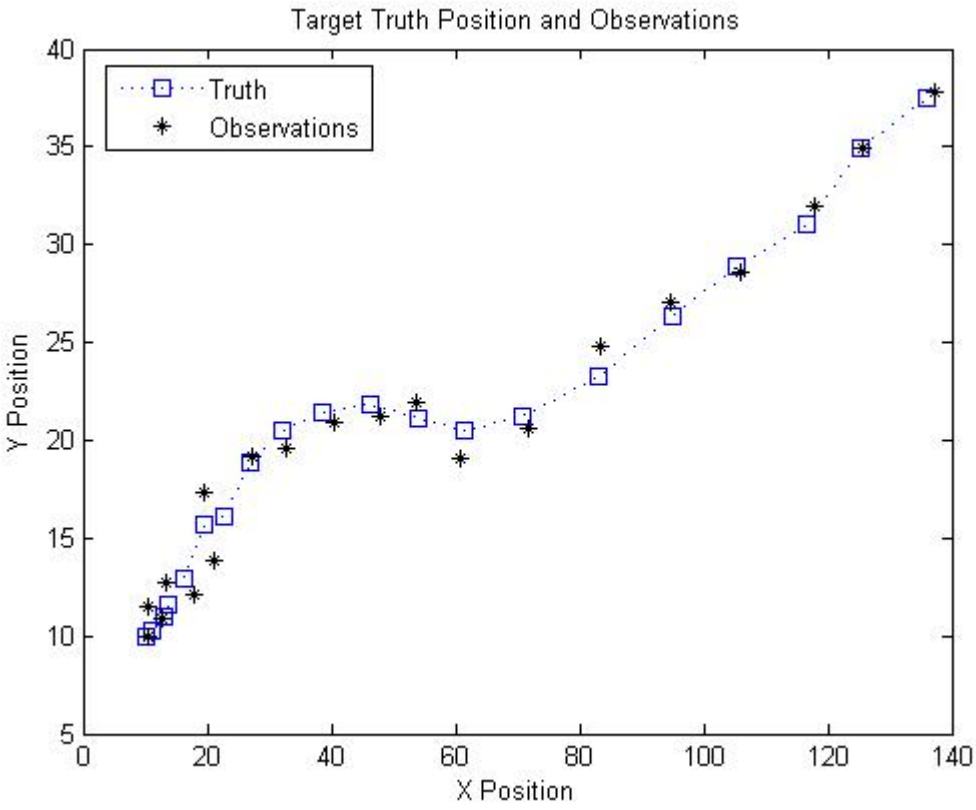

Figure 5-1: Target truth data

Kalman and particle filtering were used to generate a set of state estimation data. Figure 5-2 shows the results of the Kalman filter applied to the measurement data. Figure 5-3 shows the results of the particle filter on the same measurement data.

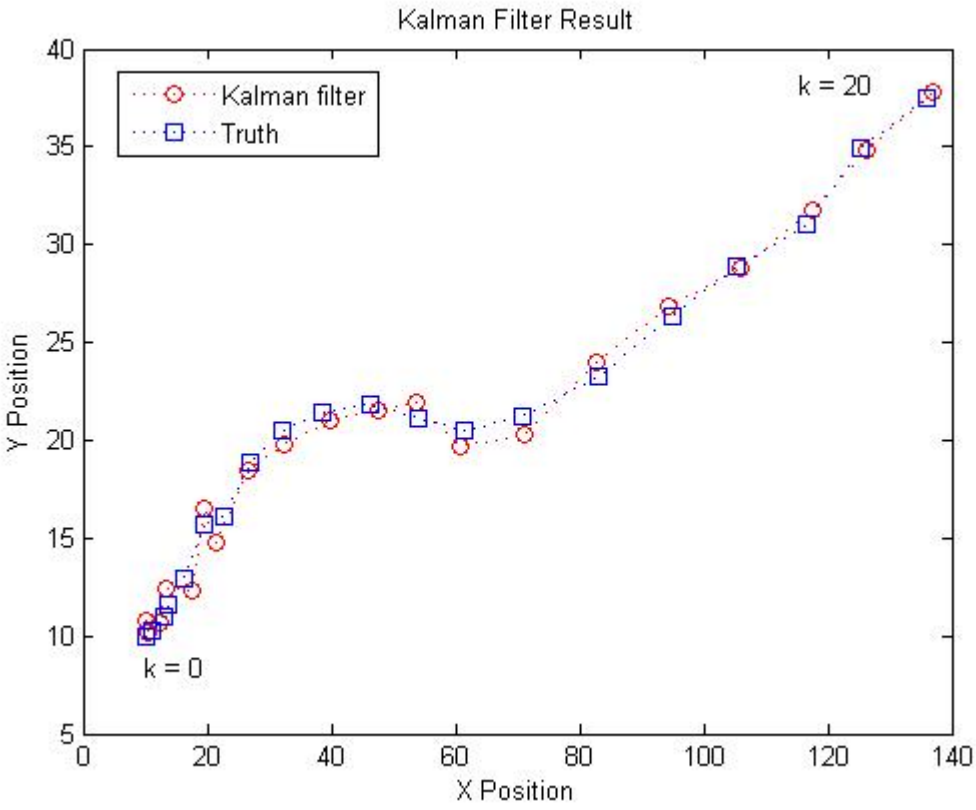

Figure 5-2: Kalman filter estimation result

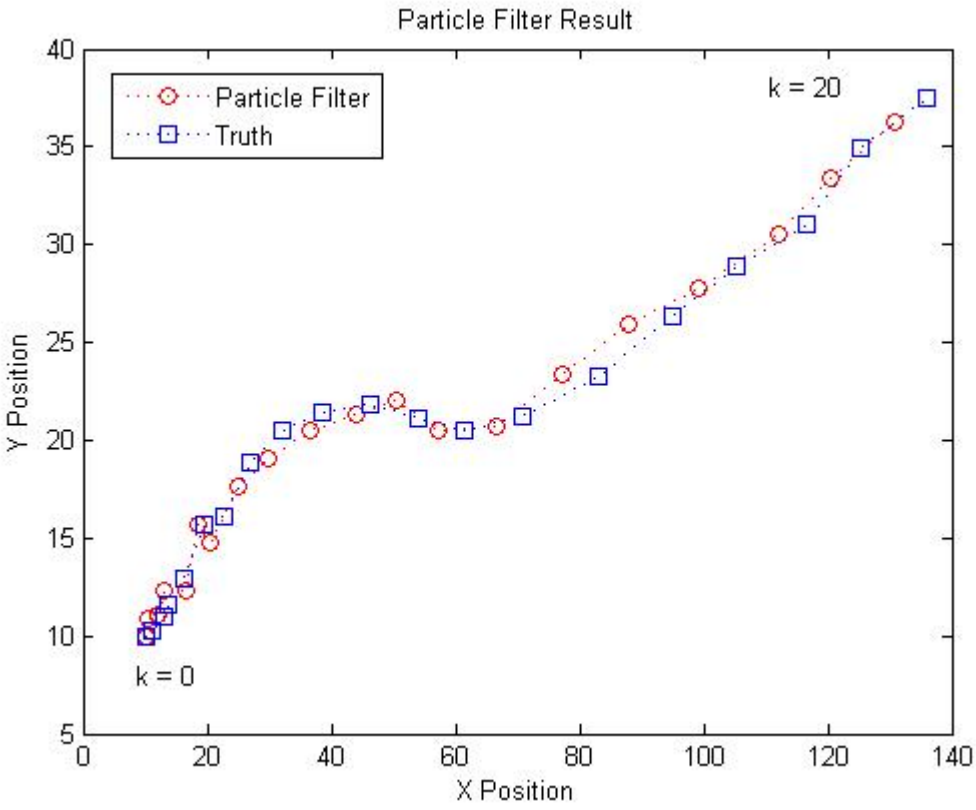

Figure 5-3: Particle filter estimation result

In order to measure the true performance of the filters, the root mean square position error (RMSPE) was computed over a set of $M = 500$ simulations of the scenario. The RMSPE is defined as:

Position error: $$PE = \sqrt{(\hat{x} - x)^2 + (\hat{y} - y)^2}$$

RMSPE: $$RMSPE = \sqrt{\frac{1}{M}\sum_{i=1}^{M} PE_i^2}$$

Filter Type	RMSPE
Kalman	1.56
Particle (100 particles)	7.34
Particle (500 particles)	7.28
Particle (1000 particles)	6.77
Particle (5000 particles)	6.61

Figure 5-4: Kalman and particle filter performance

The RMSPE for the scenario with different particle set sizes is shown in Figure 5-4. As expected, the Kalman filter outperforms the particle filter. For the case of a linear system with additive white Gaussian noise, the Kalman filter represents the optimal filtering solution. The particle filter shows only marginal improvement as the particle set size increases at the expense of extra computation.

5.2. Nonlinear Scenario

A popular nonlinear tracking problem is the angle-only tracking scenario. In this scenario, a stationary observer receives angle-only measurements of a target moving in the x-y plane. This has applications to submarine tracking using passive sonar [10] and passive range estimation of an aircraft using an electro-optic sensor such as an infrared camera [11]. Figure 5-5 shows the geometry of the observer and target in the angle-only tracking scenario.

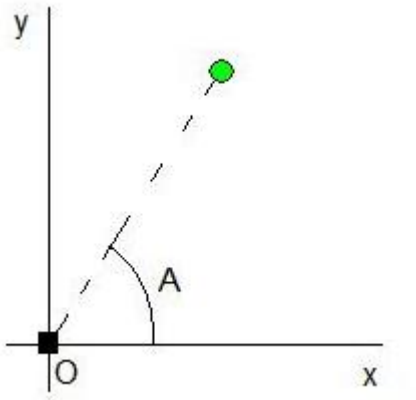

Figure 5-5: Angle-only tracking geometry

The stationary observer, O, receives nonlinear angle measurements, A, of the target position according the measurement equation:

$$z_k = \arctan(\frac{y_k}{x_k}) + v_k$$

where x_k and y_k are the target position and v_k is white Gaussian observation noise. The target moves according the constant velocity motion model. To characterize filter performance, 500 simulations were performed with process noise $\sigma_w^2 = .01$ and observation noise of $\sigma_v^2 = .01$. The initial target position was drawn at random in polar coordinates and converted to rectangular coordinates according to:

$$x_0 = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} = \begin{bmatrix} r\cos(\theta) \\ (r+\dot{r})\cos(\theta+\dot{\theta}) - r\cos(\theta) \\ r\sin(\theta) \\ (r+\dot{r})\sin(\theta+\dot{\theta}) - r\sin(\theta) \end{bmatrix}$$

where the polar coordinates are drawn from:

$$\theta \sim U[-\pi, \pi]$$
$$\dot{\theta} \sim N(0, 0.01)$$
$$r \sim N(1, 0.01)$$
$$\dot{r} \sim N(1, 0.0001)$$

A single simulation is shown in Figure 5-6. The scenario lasts for 24 time steps. Figure 5-7 shows the position estimates for the EKF and particle filter with N = 1000 particles. Figure 5-8 and Figure 5-9 show the angle and range estimates of the two filters.

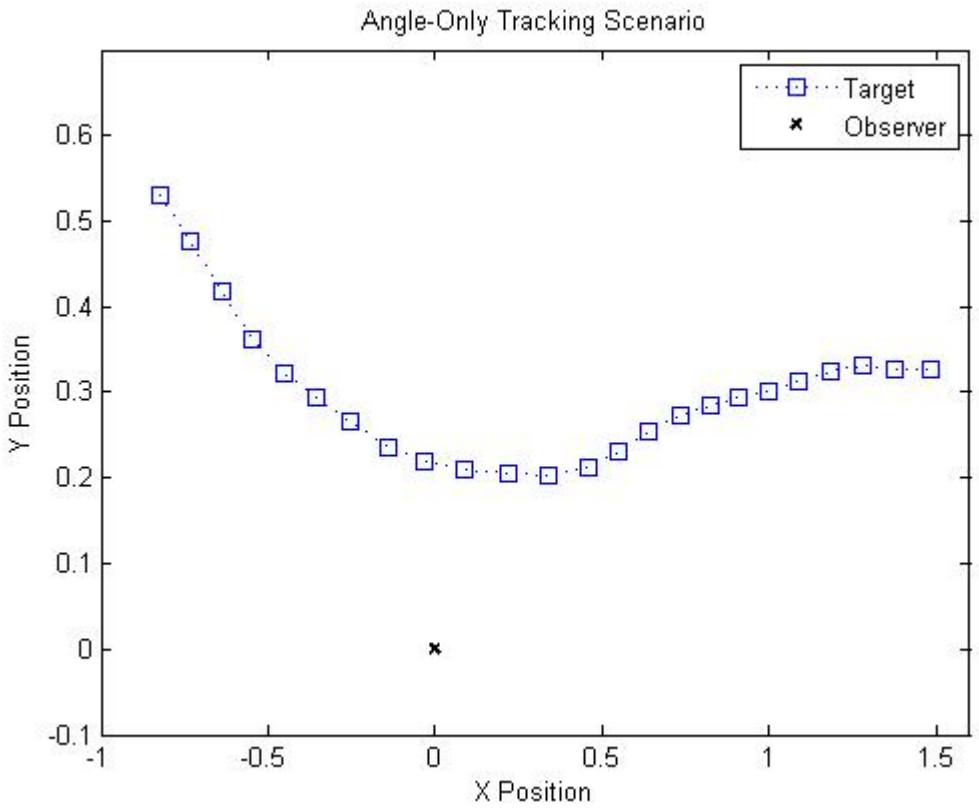

Figure 5-6: Angle-only tracking scenario

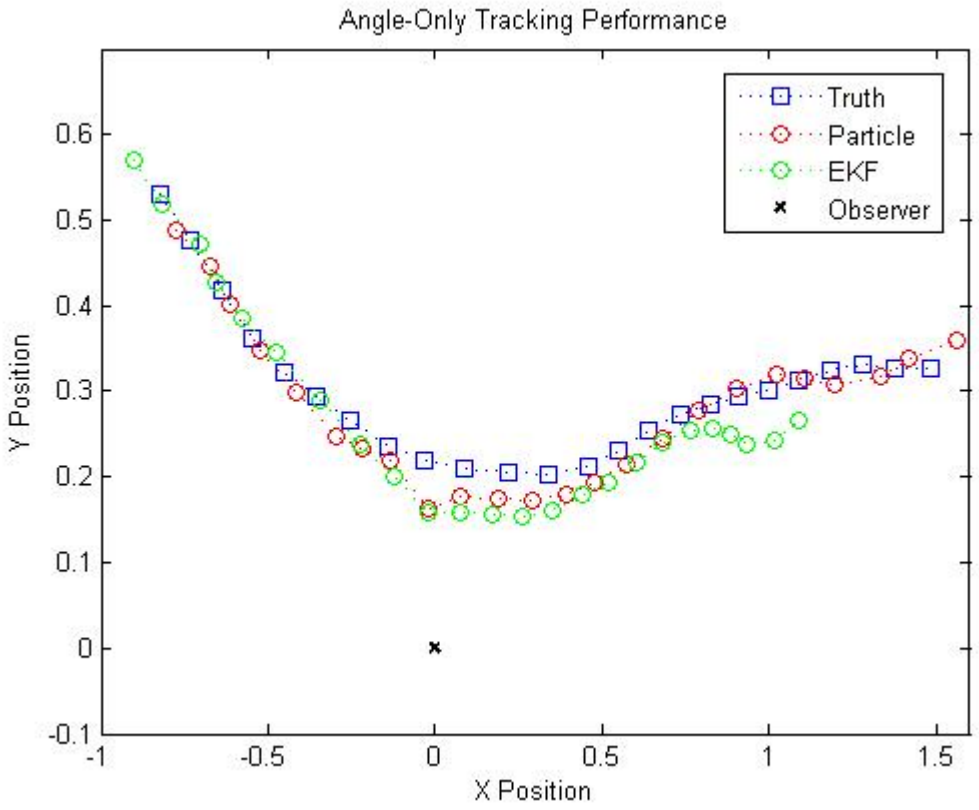

Figure 5-7: EKF and particle filter results for angle-only scenario

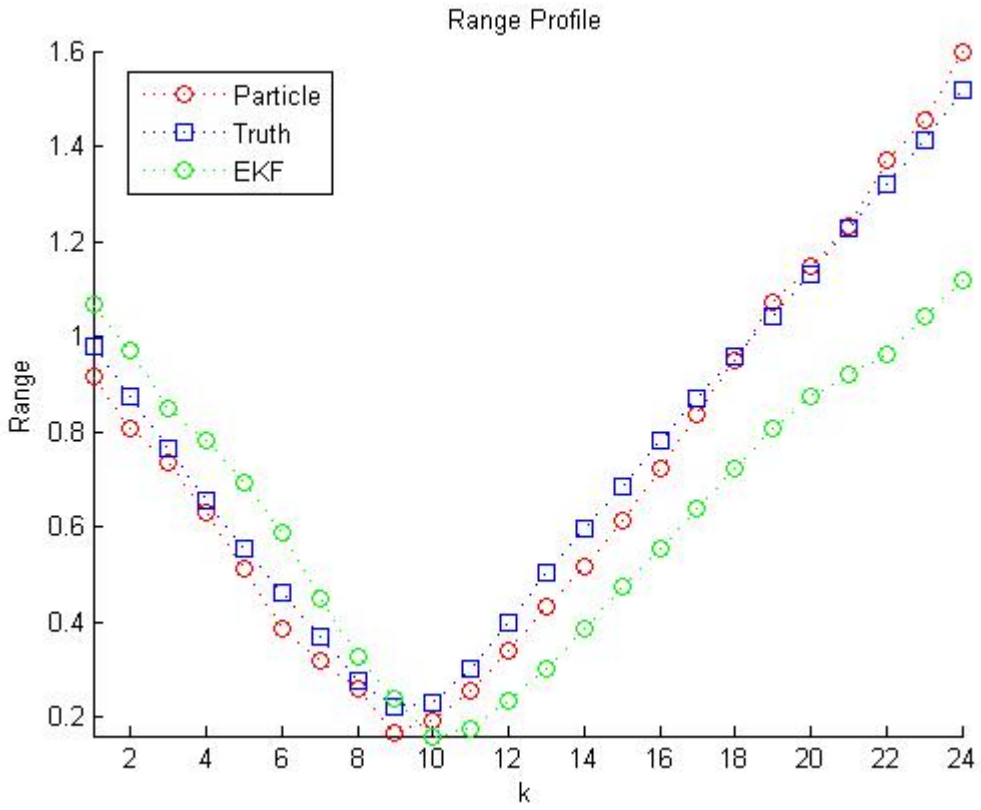

Figure 5-8: EKF and particle filter range estimates

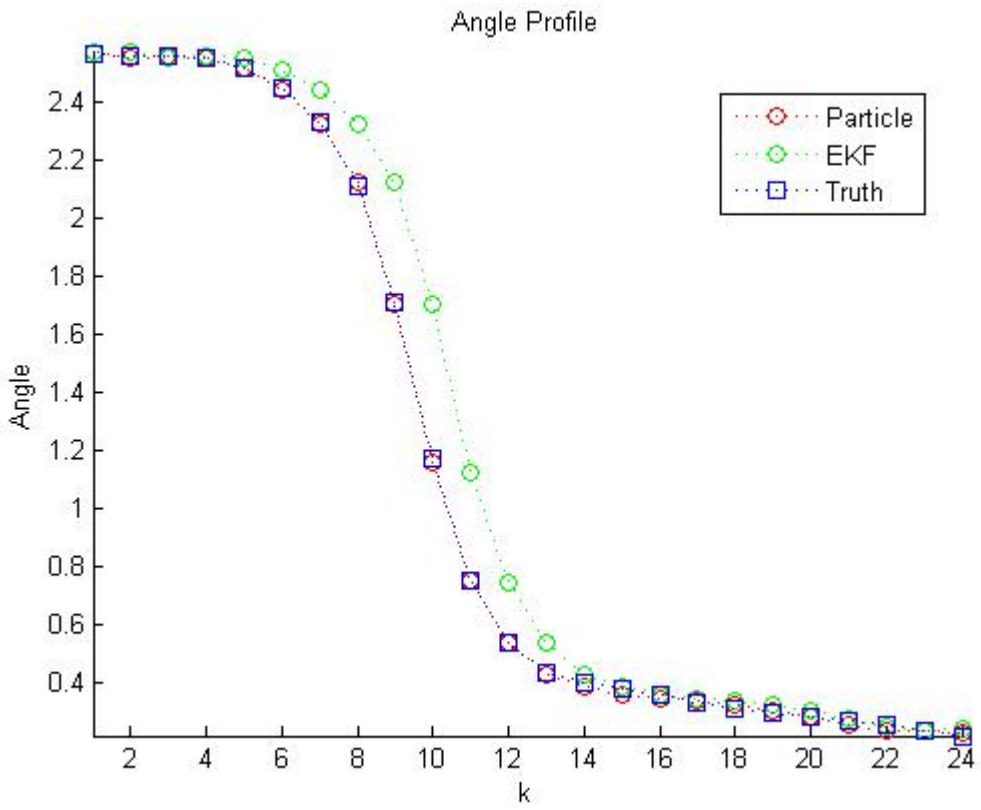

Figure 5-9: EKF and particle filter angle estimates

The RMSPE was computed over the 500 simulations and the results are shown in Figure 5-10. The particle filter clearly outperforms the EKF in this nonlinear scenario. For certain target motion trajectories, the linearization step in the EKF algorithm causes a large error in the position estimate and results in poor filter performance.

Filter Type	RMSPE
EKF	12.8
Particle (500 particles)	0.71
Particle (1000 particles)	0.62
Particle (2000 particles)	0.51

Figure 5-10: EKF and particle filter performance

6. Point Target Modeling

Most tracking applications involve the use of measurements received from electro-optic (EO) sensors and advanced radar systems such as ground moving target indicator (GMTI). In the absence of real-world measurement data, modeling and simulation of measurement data can be a very useful tool for evaluating tracking algorithm performance. EO sensor data, in the form of a series of images, can be generated with the use of a point target and simple sensor model.

6.1. Sensor Model

A typical EO sensor such as a forward-looking infrared camera observes a region in the x-y plane and returns an image of N x M pixels. Each pixel represents a region in the plane of size $\Delta \mathrm{x} \Delta$ and represents an intensity value. If a point target is present in an image, the sensor will spread the contribution of the target's intensity to surrounding pixels according to the sensor's point spread function. In practice, the point spread function is commonly represented by a 2-D Gaussian density with circular symmetry.

For a point target located at position (*x,y*) with intensity *I*, the spread contribution to pixel (*i,j*) will be:

$$S(i,j) = \frac{I\Delta^2}{2\pi\sigma_{PSF}^2} \exp\left[-\frac{(x - i\Delta)^2}{2\sigma_{PSF}^2} - \frac{(y - i\Delta)^2}{2\sigma_{PSF}^2} \right]$$

where σ_{PSF}^2 is a blurring parameter. This has the effect of blurring point targets into neighboring pixels.

6.2. Simulating Point Targets

Figure 6-1 shows the effect of the sensor model on three point targets with different blurring parameter σ_{PSF}^2 values. The far left target has $\sigma_{PSF}^2 = 4$, the middle target has $\sigma_{PSF}^2 = 1$, and the far right target has no blurring. In all cases, $\Delta = 1$ and I = 200.

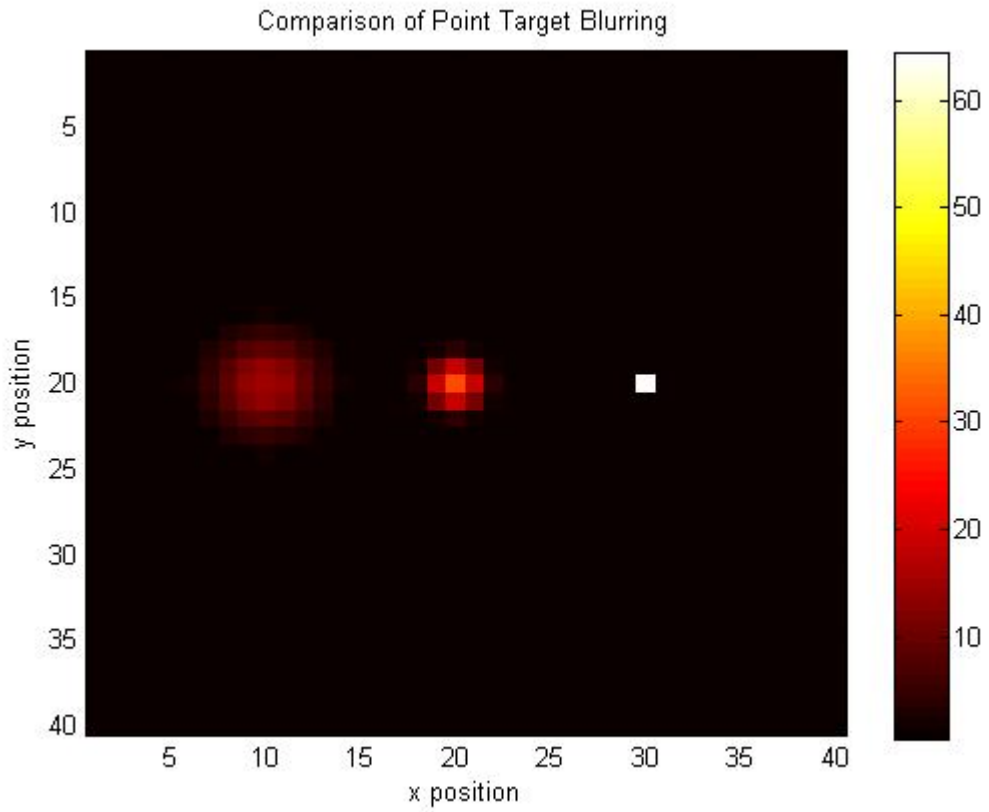

Figure 6-1: Point target blurring

6.3. Simulation Noisy Data

In order to generate realistic EO sensor, white Gaussian noise must be added to the images along with point targets. We can then define the peak pixel signal-to-noise ratio (dB) as:

$$SNR = 20\log_{10}\left[\frac{I\Delta^2}{2\pi\sigma_{PSF}^2\sigma_N}\right]$$

where σ_N is the standard deviation of the additive noise in the image.

Figure 6-2, Figure 6-3, and Figure 6-4 show simulated noisy data at various SNR levels with the middle point target of Figure 6-1 added. As the SNR level approaches 10 dB, the target becomes indistinguishable from the background noise.

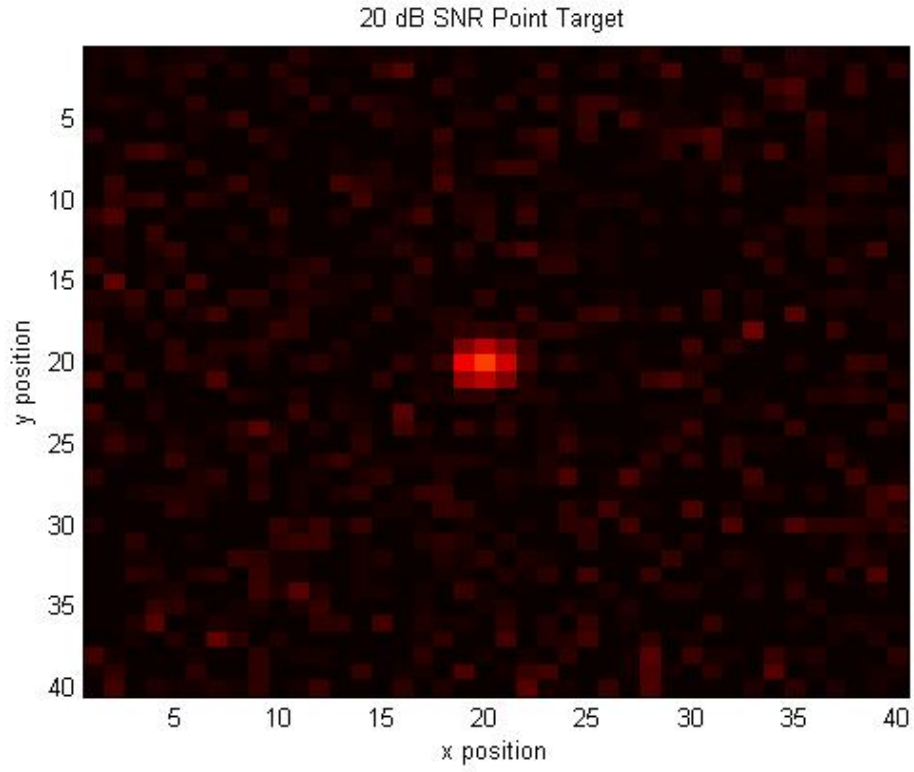

Figure 6-2: 20 dB SNR Target

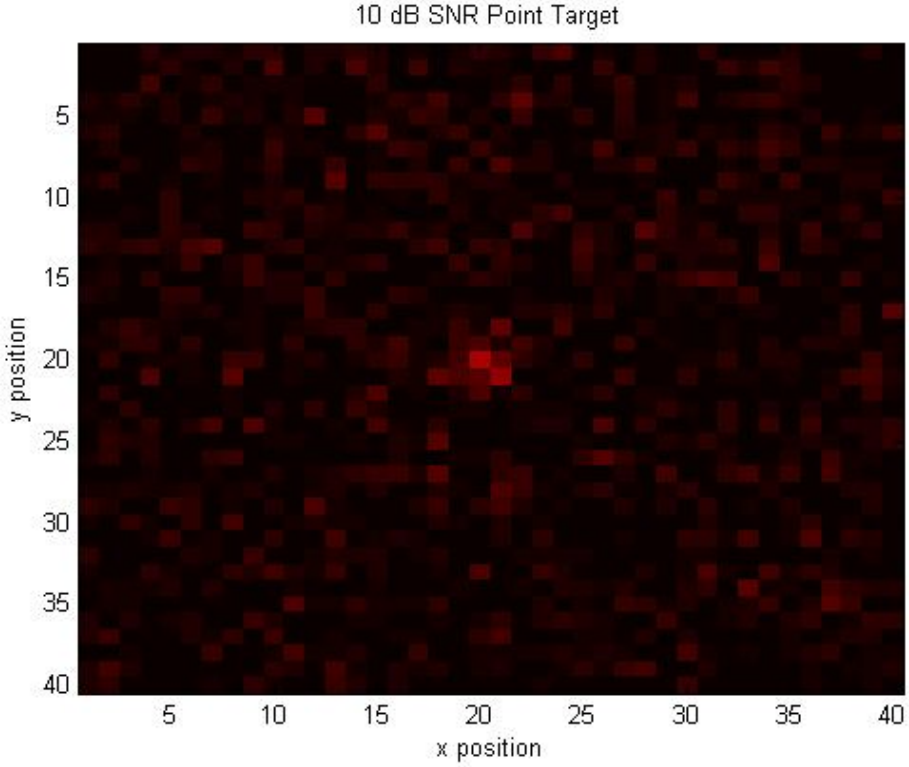

Figure 6-3: 10 dB SNR Target

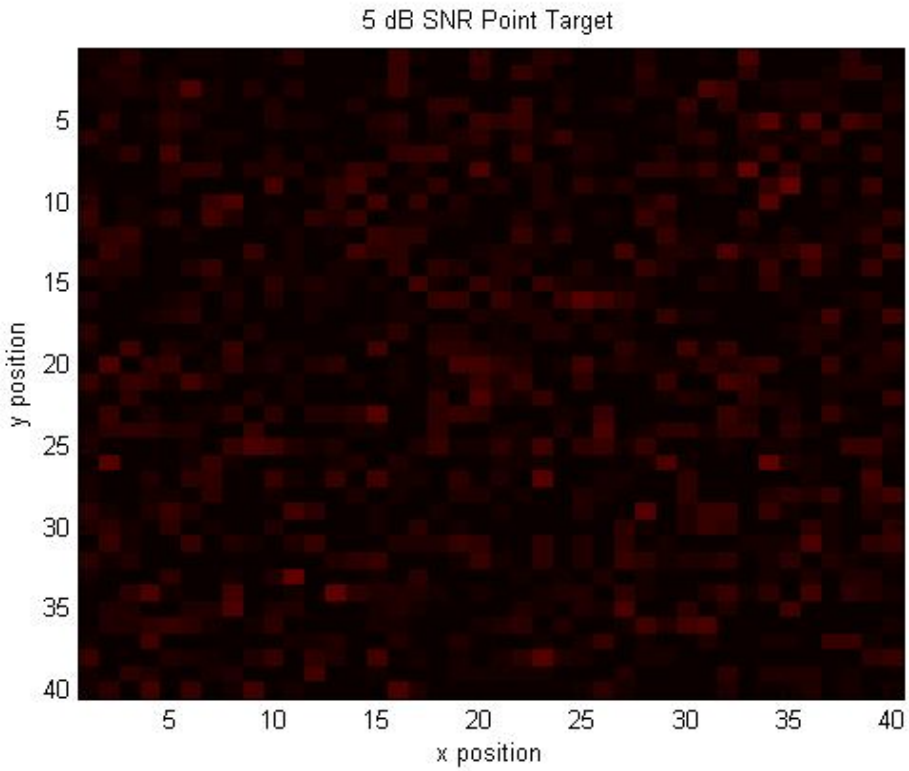

Figure 6-4: 5 dB SNR Target

7. Tracking Stealthy Targets

Tracking moving targets in noisy EO sensor imagery poses two problems, target detection and then tracking. Targets with very low SNR are hidden in the background image noise and cannot be reliably detected by thresholding the image data. Particle filtering provides a unique approach to detection and tracking of such targets known as Track-Before-Detect (TBD) [11]. In this approach, we can exploit the large particle set to search the image space for potential targets and provide a track if a target appears.

7.1. Particle Filter Algorithm for Track-Before-Detect

The standard SIR particle filter algorithm can be adapted to estimate a target's presence as well as its kinematic state [12]. The target state at time step k is modified to include intensity I_k and existence E_k variables.

$$x_k = \begin{bmatrix} x_k \\ \dot{x}_k \\ y_k \\ \dot{y}_k \\ I_k \\ E_k \end{bmatrix}$$

The target intensity I_k is an estimate of the target signal strength in the image. The target existence E_k is a binary variable with $E_k=1$ meaning target is present and $E_k=0$ meaning target is not present.

Target existence is modeled as a two-step Markov chain using the existence variable E_k. At any time, a target may appear or disappear from the

surveillance scene. This behavior is modeled with transitional probabilities P_b (probability of birth), P_d (probability of death), P_a (probability of staying alive), and P_r (probability of remaining absent) such that:

$$P_b = P\{E_k = 1 \mid E_{k-1} = 0\}$$
$$P_d = P\{E_k = 0 \mid E_{k-1} = 1\}$$
$$P_a = 1 - P_d$$
$$P_r = 1 - P_b$$

Given a series of sensor images, the Track-Before-Detect algorithm is as follows: For each image at time step *k*:

1. Randomly select P_d of the particles with $E_{k-1} = 1$. Assign $E_k = 0$ to this set. These particles have "died".

2. Randomly select P_b of the particles with $E_{k-1} = 0$. Assign $E_k = 1$ to this set. These particles are "born". Choose these particle states according to the birth proposal density function.

3. The remaining particles with $E_{k-1} = 1$ that were not chosen in step 1 are "alive" particles. Pass each sample through the system model as in Step 1 of the SIR filter algorithm of Section 3.2.

4. For each particle, compute the normalized weight as in Step 2 of the SIR filter algorithm of Section 3.2.

5. Resample the particles as in Step 3 of the SIR filter algorithm of Section 3.2.

6. Compute the probability of existence estimate PE_k and state estimate

$$\hat{x}_k^{MEAN}$$

7.2. Birth Proposal Density Function

The birth proposal density function determines how the state of each "born" particle is chosen in Step 3 of the TBD algorithm. The simplest method is a blind proposal that relies upon no prior knowledge of the target existence or state [14]. Particle positions x_k and y_k are uniformly distributed about the image. Particle velocities $\dot{x}_k$ and $\dot{y}_k$ are uniformly distributed between chosen minimum and maximum velocity values v_{min} and v_{max}. Particle intensity I_k is uniformly distributed between chosen minimum and maximum intensity values I_{min} and I_{max}.

7.3. Weight Calculation

The measurement for each time step k consists of a sensor image of size N x M. Each pixel intensity value $z_{i,,j}$, where i = 1,…,N and j = 1,…,M, in the image can be modeled as:

$$z_{i,j} = h_{i,j}(x_k) + v_{i,j} \quad \text{when target present}$$

$$z_{i,j} = w_{i,j} \quad \text{when target not present}$$

where $h_{i,,j}(x_k)$ is the contribution of the target and $v_{i,,j}$ is the zero-mean Gaussian measurement noise at pixel *i,j*. The function $h_{i,,j}(x_k)$ is the point spread function from Section 6.1 and is defined as:

$$h_{i,j}(x_k) = \frac{I_k \Delta^2}{2\pi\sigma_{PSF}^2} \exp\left[-\frac{(x_k - i\Delta)^2}{2\sigma_{PSF}^2} - \frac{(y_k - i\Delta)^2}{2\sigma_{PSF}^2} \right]$$

For each image pixel $z_{i,j}$ and particle state x_k^n, the likelihood that a target is present is defined as:

$$l_{i,j}(z_{i,j} \mid x_k^n) = \frac{p(signal) + p(noise)}{p(noise)} = \exp\left(- \frac{h_{i,j}(x_k^n)\left[h_{i,j}(x_k^n) - 2z_{i,j}\right]}{2\sigma^2} \right)$$

where σ^2 is the measurement noise variance. Here for simplicity we assume that if a target is present, it will contribute to a 3x3 region *C* of pixels around its location. We can then define the weight for each particle x_k^n to be:

$$w_k^n = \prod_C l_{i,j}(z_{i,j} \mid x_k^n) \quad \text{if } E_k = 1$$

$$w_k^n = 1 \qquad \text{if } E_k = 0$$

If a particle is "alive", it's weight is the product of the likelihood of the 3x3 region of pixels around its location. If a particle is "dead", its weight is 1.

7.4. Probability of Existence and State Estimate Calculation

The probability of existence at time step *k* is the ratio of "alive" particles to total number of particles *N* and is computed using the existence variable E_k:

$$PE_k = \frac{1}{N} \sum_{n=1}^{N} E_k^n$$

The state estimate $\hat{x}_k^{MEAN}$ is similar to the state estimate of the SIR filter algorithm in Section 3.2 and is the mean of all "alive" particles:

$$\hat{x}_k^{MEAN} = \frac{\sum_{n=1}^{N} x_k^n E_k^n}{\sum_{n=1}^{N} E_k^n}$$

8. Single Target Detection and Tracking Scenario

In this scenario, a target moves in the x-y plane. An EO sensor captures 30 measurement images of data. Each image is 50x50 pixels. The target appears at image 7 and disappears at image 22. The initial target state vector is:

$$x_0 = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ I \end{bmatrix} = \begin{bmatrix} 10 \\ 1.0 \\ 10 \\ 1.5 \\ 12.26 \end{bmatrix}$$

The target moves according to the CV model with zero mean Gaussian process noise and $\sigma_w^2 = 1$. The measurement noise is zero mean Gaussian with variance $\sigma_v^2 = 1$. The initial target intensity of 12.26 corresponds to a SNR of 12 dB. The particle filter contains N = 30,000 particles. The probability of birth and death are set to $P_b = P_d = 0.05$. The minimum and maximum particle velocity values are $v_{min} = -3$ and $v_{max} = 3$. The minimum and maximum intensity values are $I_{min} = 5$ and $I_{max} = 15$. Initially, 5% of the particles are "born" and are distributed according to the birth proposal density.

8.1. Single Simulation Results

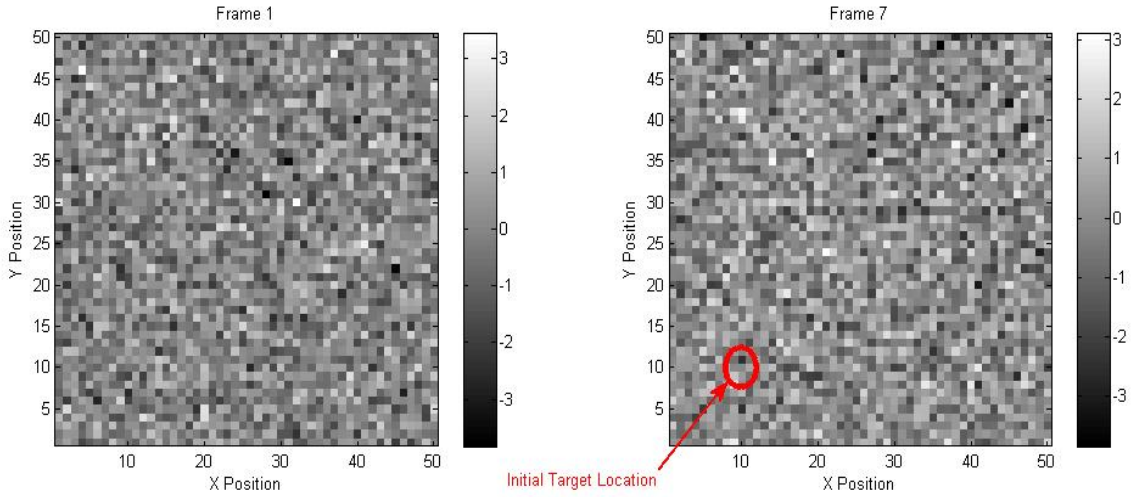

Figure 8-1: Sample image frames from scenario

Two sample image frames from a scenario simulation are shown in Figure 8-1 at different times. The target appears in frame 7 at the marked location, but is indistinguishable from the background noise.

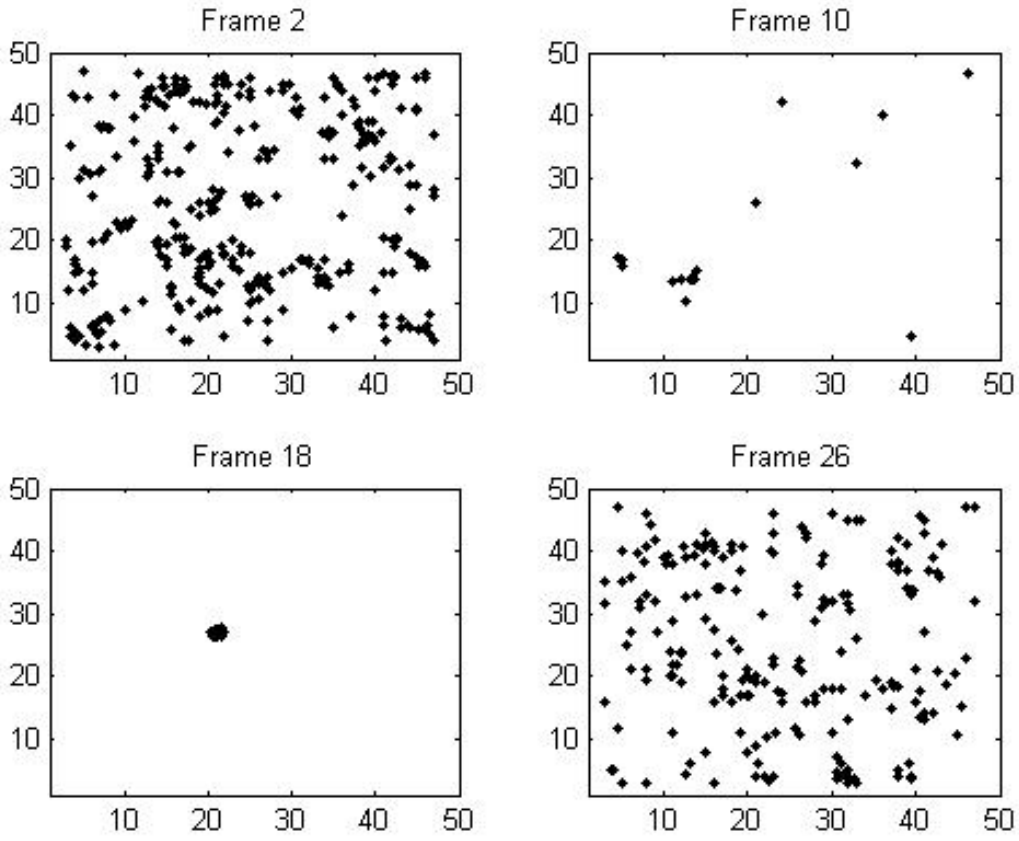

Figure 8-2: Particle locations at different frames

Figure 8-2 illustrates the progression of particle locations at various frames in the scenario. Initially, there is no target present and the particles are randomly distributed over the image. When the target appears and persists in frames 10 and 18, particles near the target location have large weights and will multiply in the resampling step. This causes the particle set to condense to the target location. When the target disappears as in frame 26, the weights of all particles are near equal and the set will be randomly distributed over the image again.

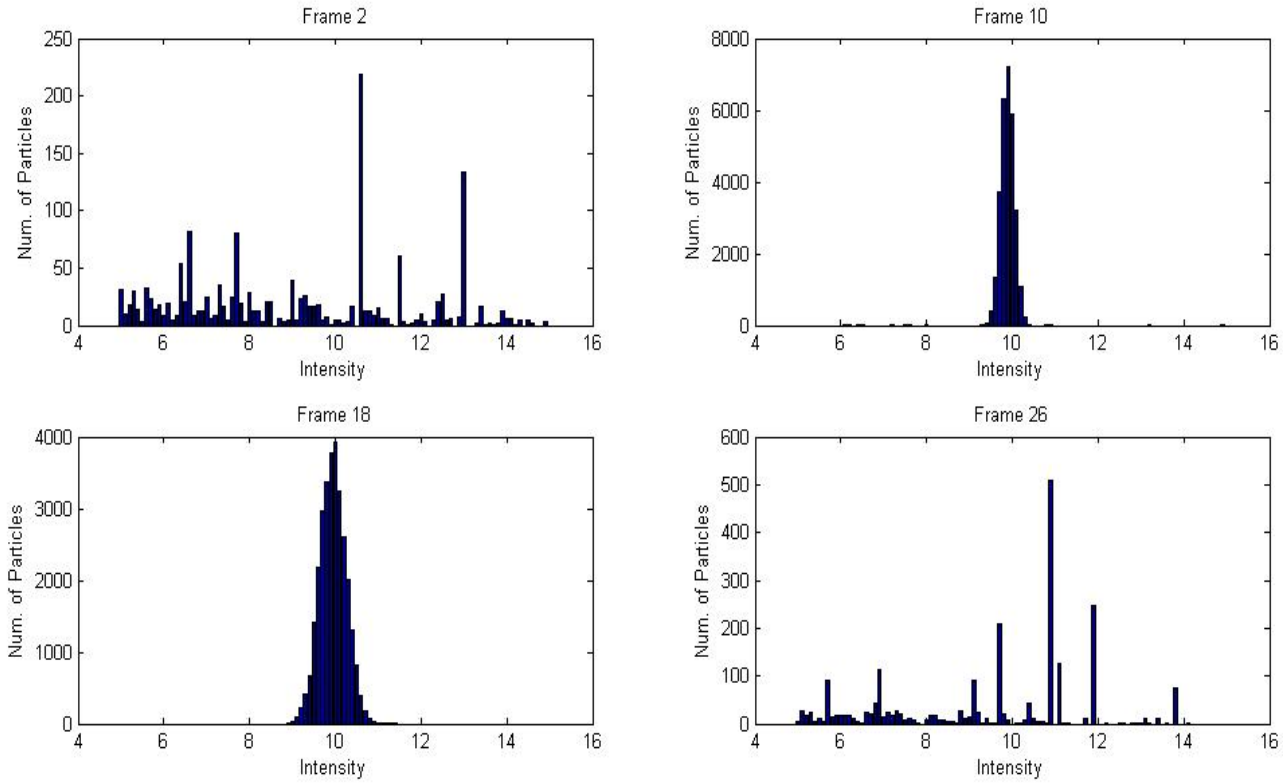

Figure 8-3: Particle intensities at different frames

Similar to the behavior shown in Figure 8-2, the particle intensities in Figure 8-3 converge to the actual target intensity when the target is present in frames 10 and 18. The particle intensities are widely distributed over the allowed range when the target is not present in frames 2 and 26.

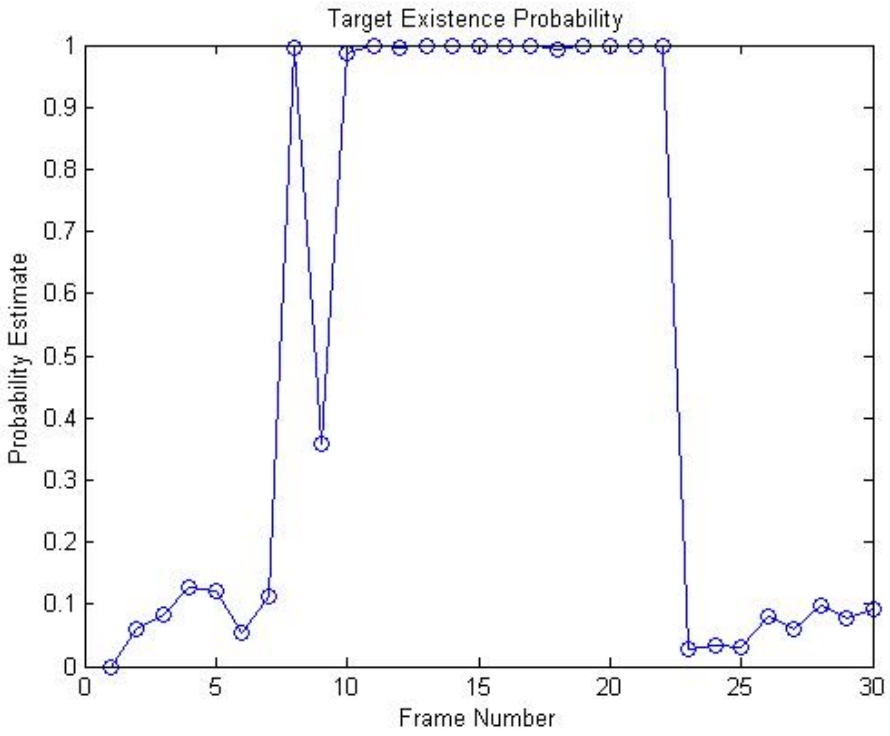

Figure 8-4: Target existence probability estimate

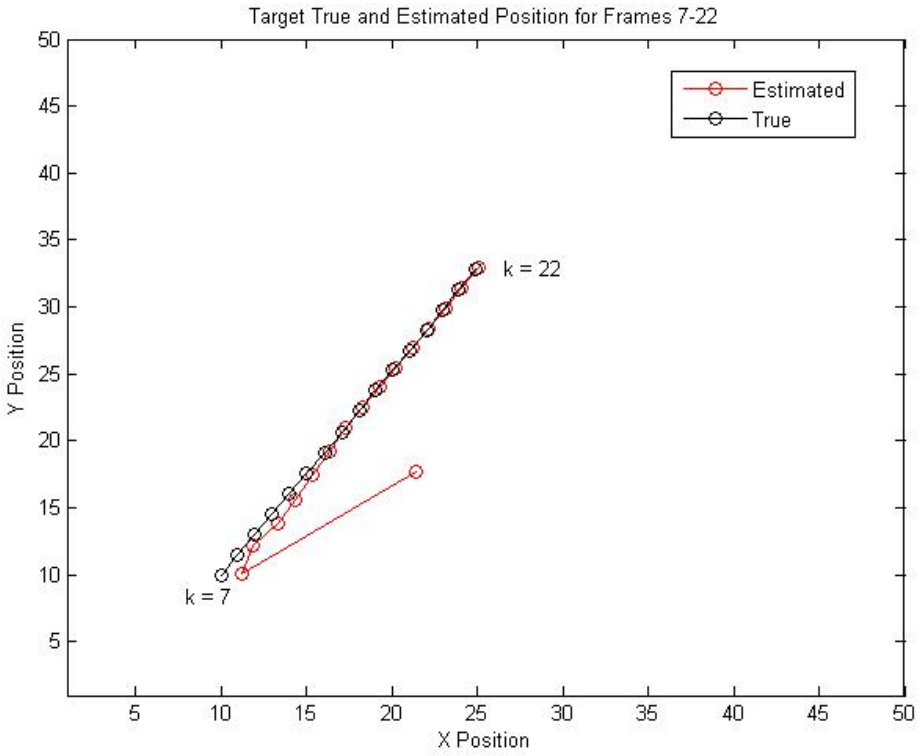

Figure 8-5: True and estimated target position

The target existence probability is shown in Figure 8-4 and the position tracking is shown in Figure 8-5. The estimated position is very close to the true target position in frames 7 through 22. The existence probability estimate provides a good indicator of target existence.

8.2. Performance Analysis – Particle Set Size

In order to determine the performance of the algorithm in tracking a single stealthy target, 100 simulation runs were performed and the results were averaged. The scenario is the same as above. The particle minimum and maximum intensity values are set to $I_{min} = 4.35$ and $I_{max} = 30.8$. These values correspond to target intensities of 3 and 20 dB respectively. The true target

intensity is unknown so a wide range of intensity values is accommodated. The particle set size was varied between 30,000 and 70,000 particles. The target intensity is 12 dB. Figure 8-6, Figure 8-7, and Figure 8-8 show the effect of varying the particle set size upon tracking performance.

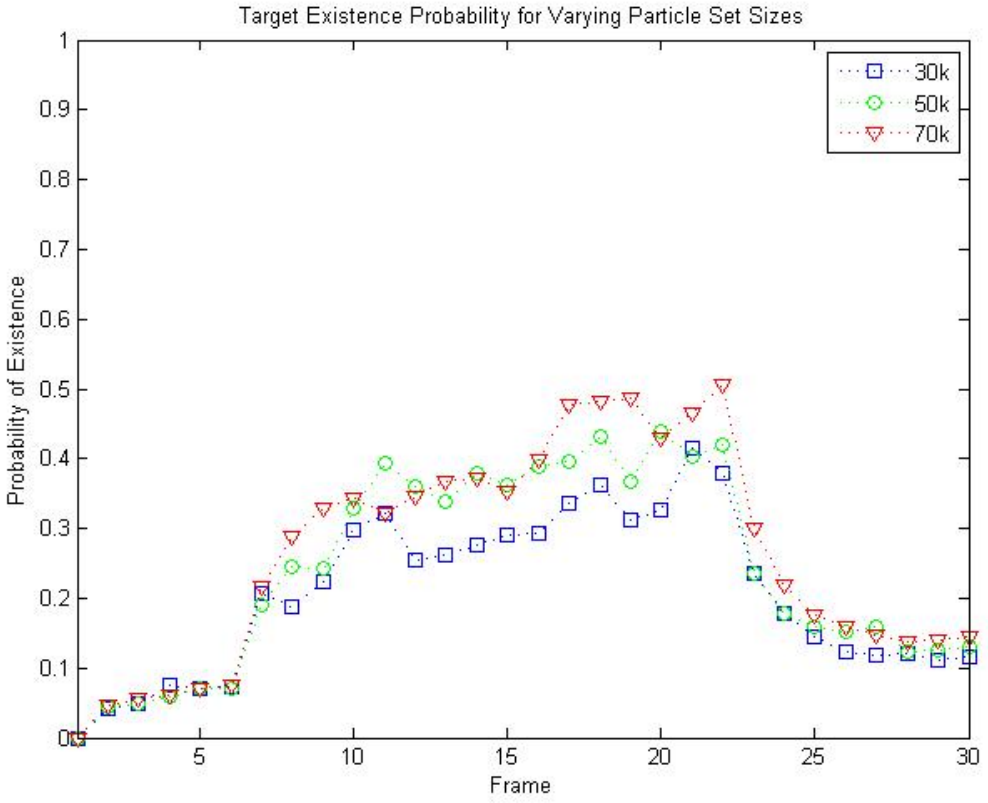

Figure 8-6: Target existence probability for various particle set sizes

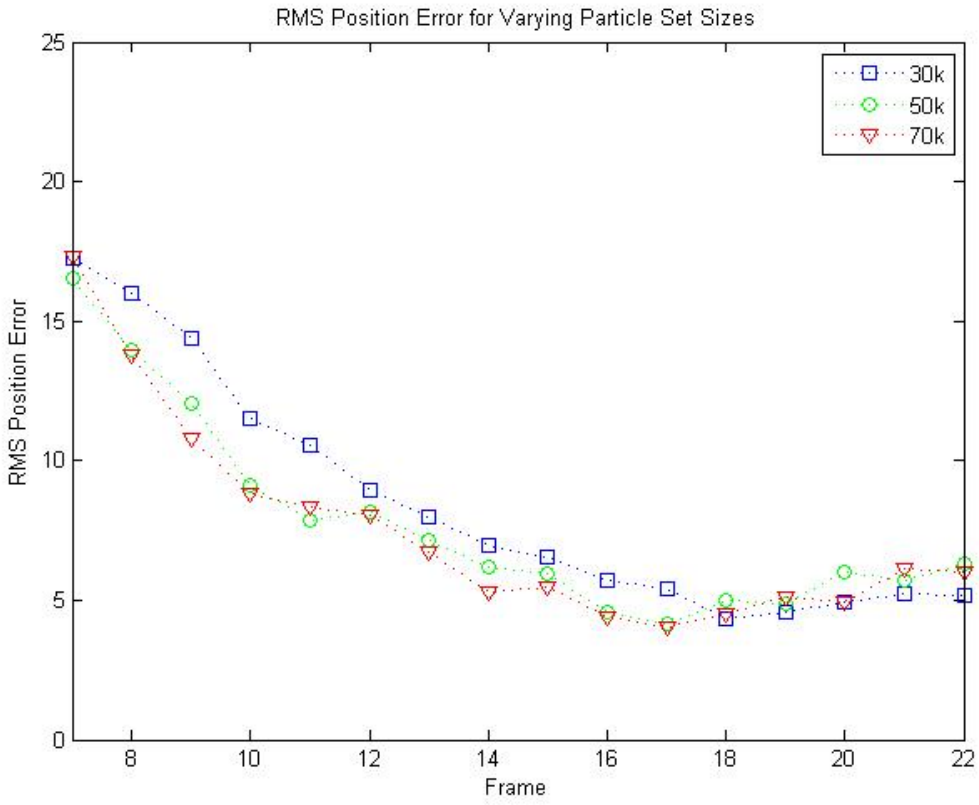

Figure 8-7: RMS position error for various particle set sizes

Choosing a particle set size of 50,000 particles produces the best compromise between target detection and target tracking performance. In Figure 8-6, not much is gained by increasing the particle set size to 70,000 particles at the expense of computation time. A threshold value of 0.2 would provide an accurate measure of target existence. The RMS position error in Figure 8-7 and the RMS intensity error in Figure 8-8 show no advantage in increasing the particle set size beyond 50,000.

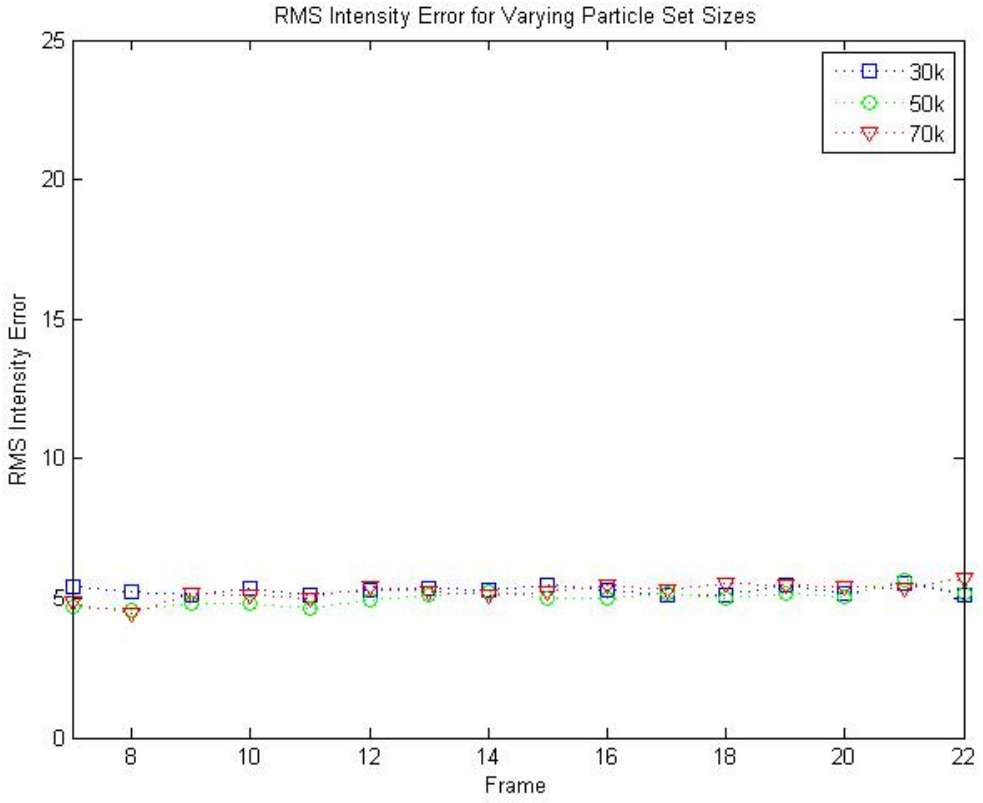

Figure 8-8: RMS intensity error for various particle set sizes

8.3. Performance Analysis – Target SNR

Using the same scenario as the previous section and choosing a particle set size of 50,000 as a benchmark, the target SNR was varied between 9 dB and 15 dB to determine the performance of the algorithm for strong and weak targets. The results are shown in Figure 8-9, Figure 8-10, and Figure 8-11.

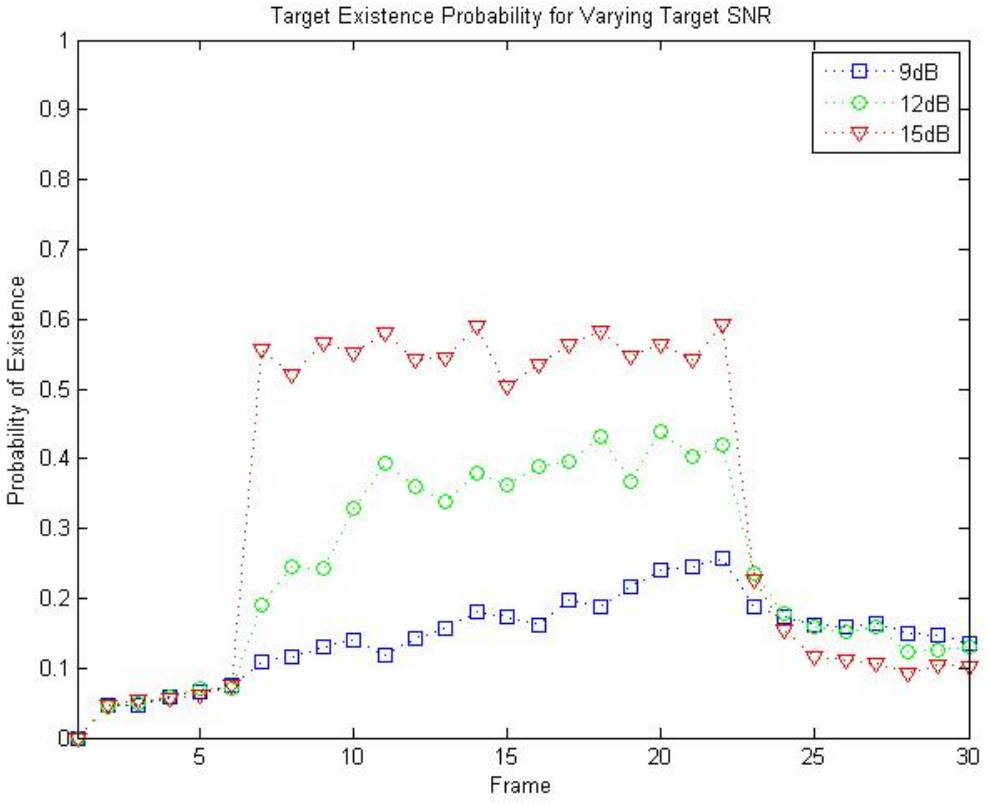

Figure 8-9: Target existence probability for various target SNR levels

The algorithm performs poorly for targets below 12 dB. The probability of existence estimate increases slightly when the target appears in frame 7 for the 9 dB target. This increase is too small to reliable determine if a target is present. As expected, as target SNR increases, the existence estimate jumps sharply when the target appears and disappears.

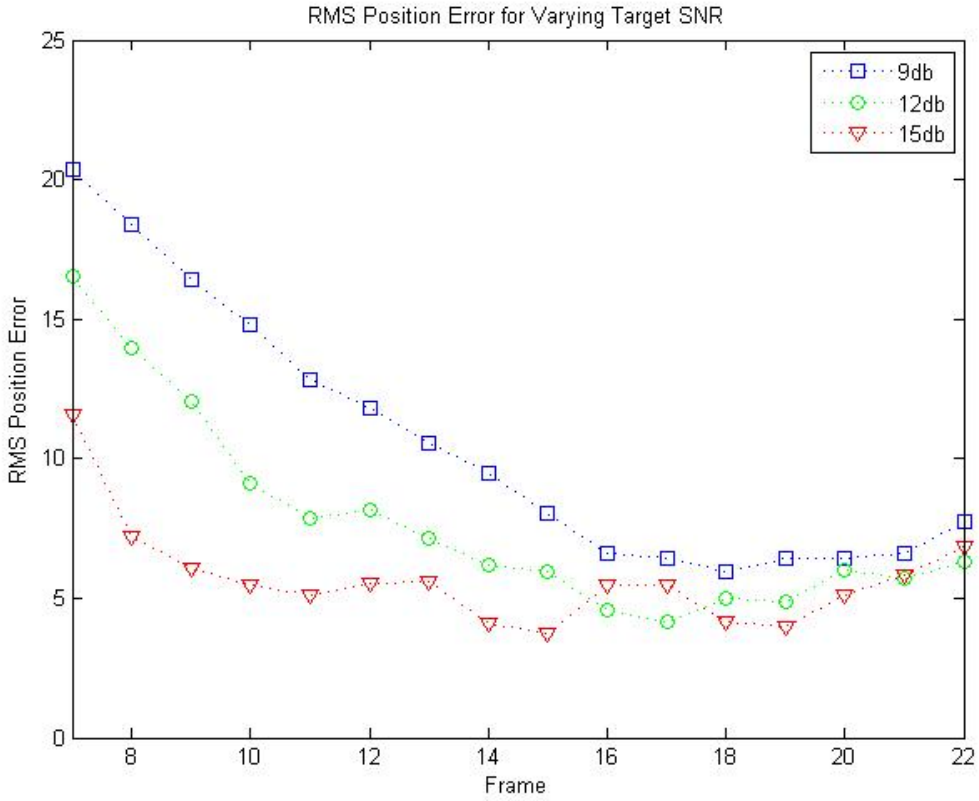

Figure 8-10: RMS position error for various target SNR levels

The 15 dB target provides the lowest tracking error over the time span of frame 7 to 22. All three targets converge to similar RMS position errors in the last few frames.

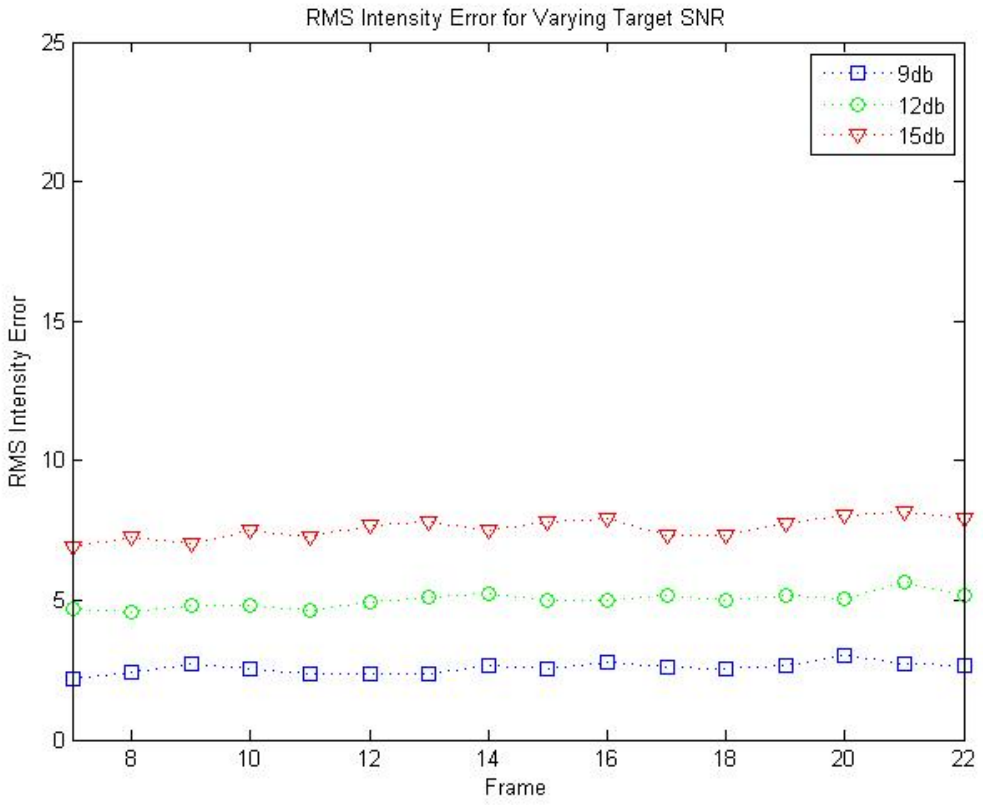

Figure 8-11: RMS intensity error for various target SNR levels

The 9 dB target provided the lowest RMS intensity error. This is unexpected as the algorithm makes no assumptions about target SNR.

8.4. Improved Particle Filter Algorithm for Track-Before-Detect

The birth proposal density function described in Section 7.2 does not make use of the image data to determine where new particles are placed within the image. This function is blind to the image data. When a target appears in an image, pixel values at its location are likely to be larger than the average noisy pixel value. A simple and effective way to increase filter performance is to threshold the image and place “born” particles in locations with large pixel intensity values [14].

Figure 8-12, Figure 8-13, and Figure 8-14 show the results of applying the data-driven proposal function to the single target scenario above. The filter particle set size is set to 50,000. The target SNR is 12 dB. The image data threshold is set such that "born" particles are placed in the top 25% and top 5% pixels values.

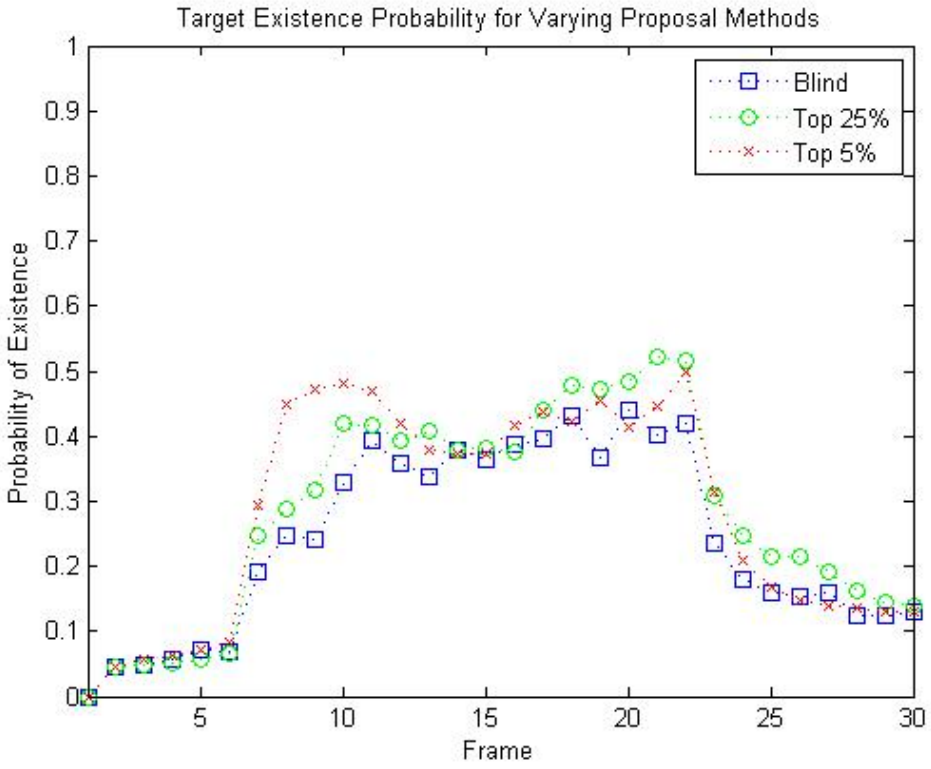

Figure 8-12: Target existence probability for various proposal methods

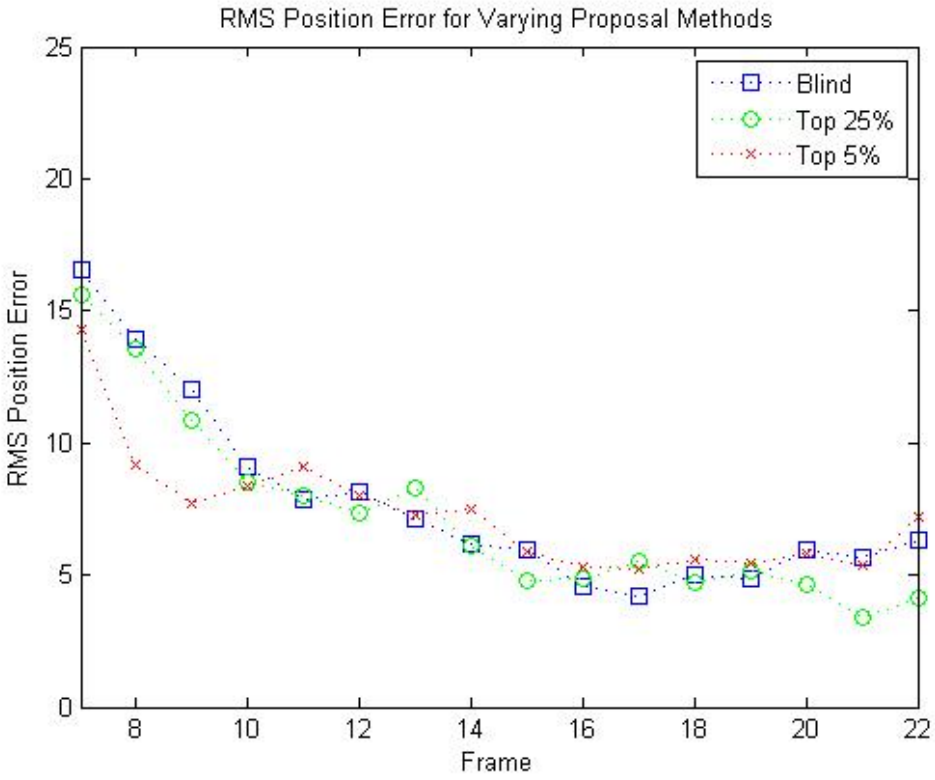

Figure 8-13: RMS position error for various proposal methods

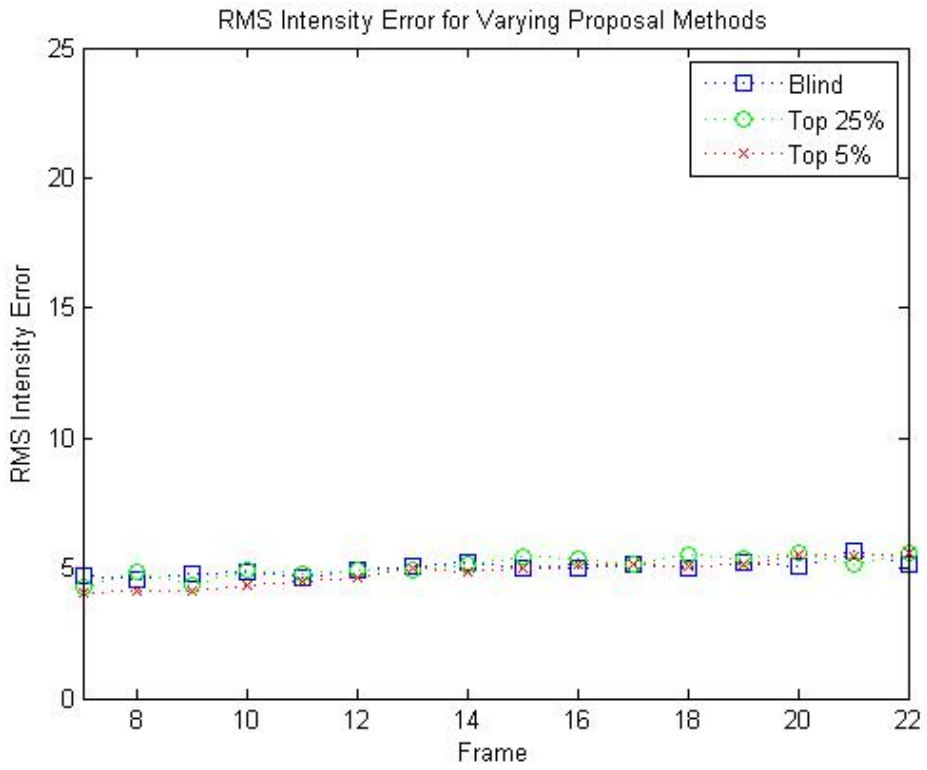

Figure 8-14: RMS intensity error for various proposal methods

The 5% proposal method showed improved detection performance in frames 7-13 when compared to the blind method in Figure 8-12. The RMS position and intensity errors in Figures 13 and 14 do not show much improvement in the 5% proposal method over the blind and 25% method. The RMS position error is lower for the 5% proposal method during the first few frames of target existence.

9. Multiple Target Detection and Tracking

Consider a scenario in which two targets appear and disappear over a sequence of 40 image frames. The target motions are shown in Figure 9-1. Target 1 appears in the image at frame 5 and disappears at frame 25. Target 2 appears at frame 15 and disappears at frame 35. Both targets move with constant velocity.

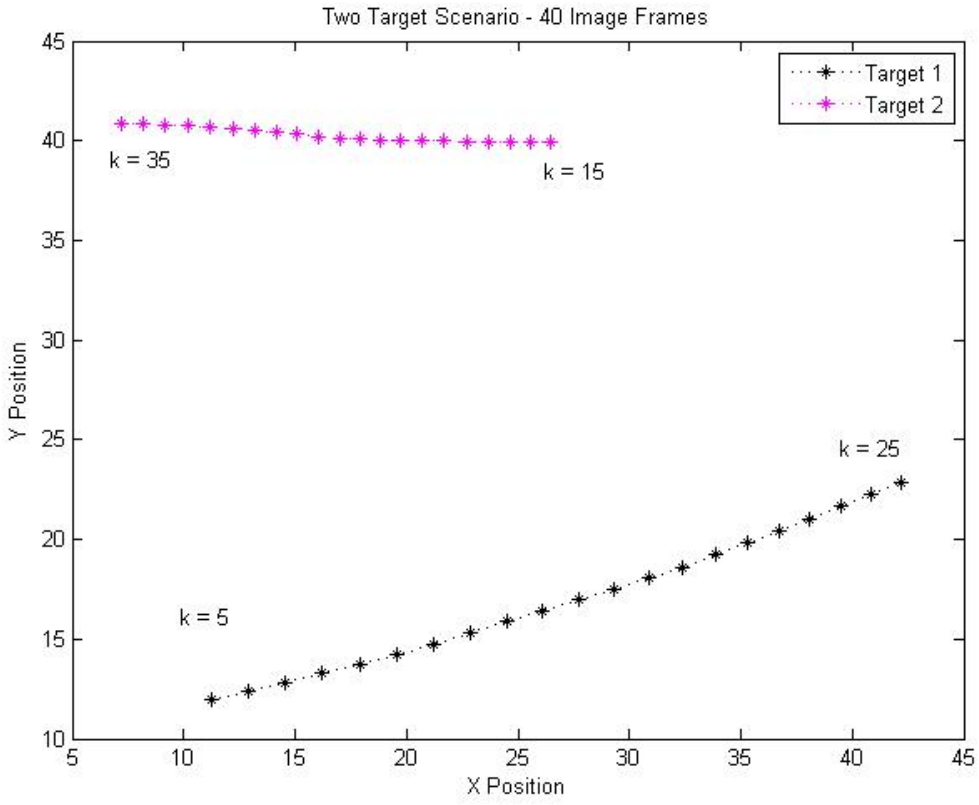

Figure 9-1: Target positions in two target scenario

The particle filter algorithm can be extended to accommodate the tracking of multiple targets by adding an extra filter for each expected target. For this two target scenario, two filters are run simultaneously on the image sequence.

9.1. Position Variance Measure

In order to prevent both filters from tracking the same target, a measure is needed to determine when a filter is tracking or searching for a target. When a target is present, the filter particle set will condense to the target location. When a target is not present, the particle set will be uniformly distributed in the image. This behavior is illustrated in Figure 9-2.

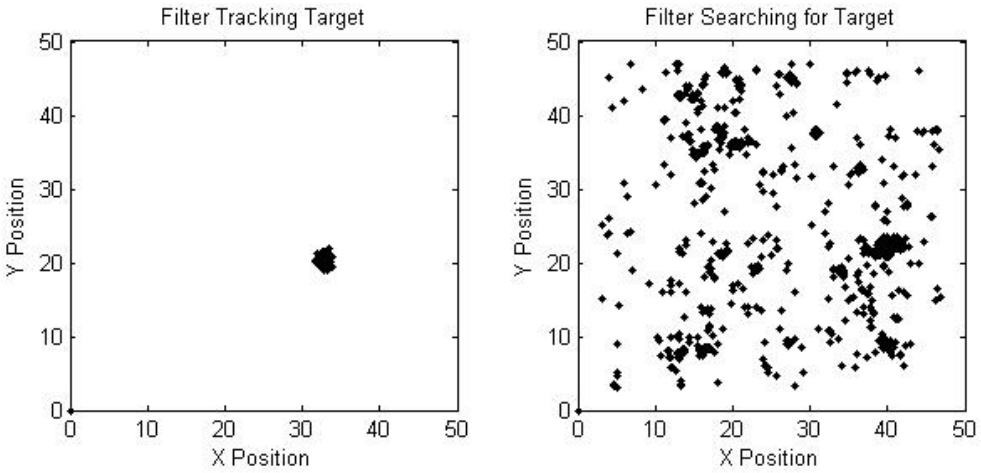

Figure 9-2: Particle locations when a target is present and absent

The position variance of the particle set provides a measure of how compact the particle set is in the image. In Figure 9-2, the position variance when the filter is tracking a target is:

$$\sigma_X^2 = 0.03$$
$$\sigma_Y^2 = 0.04$$

In contrast, the position variance when the filter is searching for a target is:

$$\sigma_X^2 = 125.50$$
$$\sigma_Y^2 = 134.30$$

If the image is 50x50 pixels in size and the particles are uniformly distributed over the image, the maximum position variance value is:

$$\sigma_X^2 = \sigma_X^2 = \frac{(50-1)^2}{12} = 200$$

Similarly, the minimum position variance value is 0 when all particles occupy the same position. A hypothesis test to determine if a filter is tracking or searching for a target is then:

$$\sigma_X^2, \sigma_Y^2 < \sigma_{THRESH}^2 \rightarrow tracking$$
$$\sigma_X^2, \sigma_Y^2 > \sigma_{THRESH}^2 \rightarrow searching$$

For the two target scenario, a suitable choice for σ_{THRESH}^2 = 100.

9.2. Track Gating

When one filter is tracking a target, most of its particles are condensed to a region around the target location. The particle sets from other filters must not occupy this region. Otherwise, those filters would track the same target. A simple method to accomplish this is if one filter is tracking a target, define a circular region of radius *R* centered on the mean. This is called the *track gate*. Remove particles from all other filters that are within this track gate.

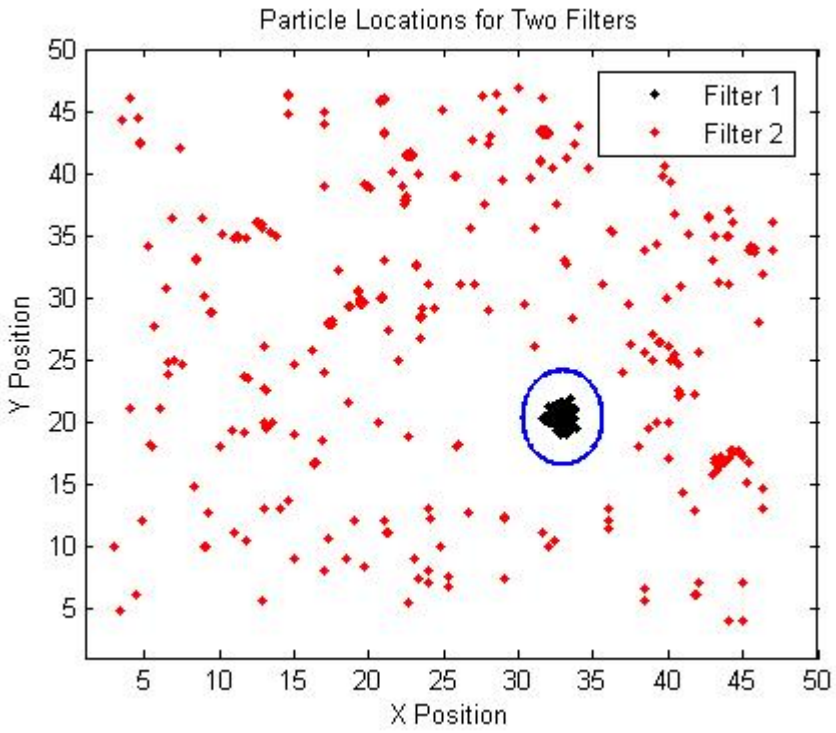

Figure 9-3: Particle locations for two filters

The method is illustrated in Figure 9-3. Filter 1 is tracking a target and its track gate is shown in blue. All particles from Filter 2 are removed Filter 1's track gate, preventing Filter 2 from tracking the same target as Filter 1.

9.3. Multiple Target Particle Filter Algorithm

We can now define the Track-Before-Detect algorithm for multiple targets as:

For each image at time step *k*:

1. For each filter, perform steps 1-3 in Section 7.1.
2. For each filter, compute the position variance and determine if the filter is searching or tracking.

3. For each filter that is tracking, eliminate all particles from other filters within its track gate.

4. For each particle, compute the normalized weight as in Step 2 of the SIR filter algorithm of Section 7.1.

5. Resample the particles as in Step 3 of the SIR filter algorithm of Section 3.2.

6. Compute the probability of existence estimate PE_k and state estimate $\hat{x}_k^{MEAN}$

9.4. Two Target Scenario Results

The multiple target Track-Before-Detect algorithm was applied to a single run of the two target scenario described in Figure 9-1. Both targets have a SNR of 12 dB. The filter and noise parameters are the same as the single target scenario described in Section 8 except the particle set size is 50,000.

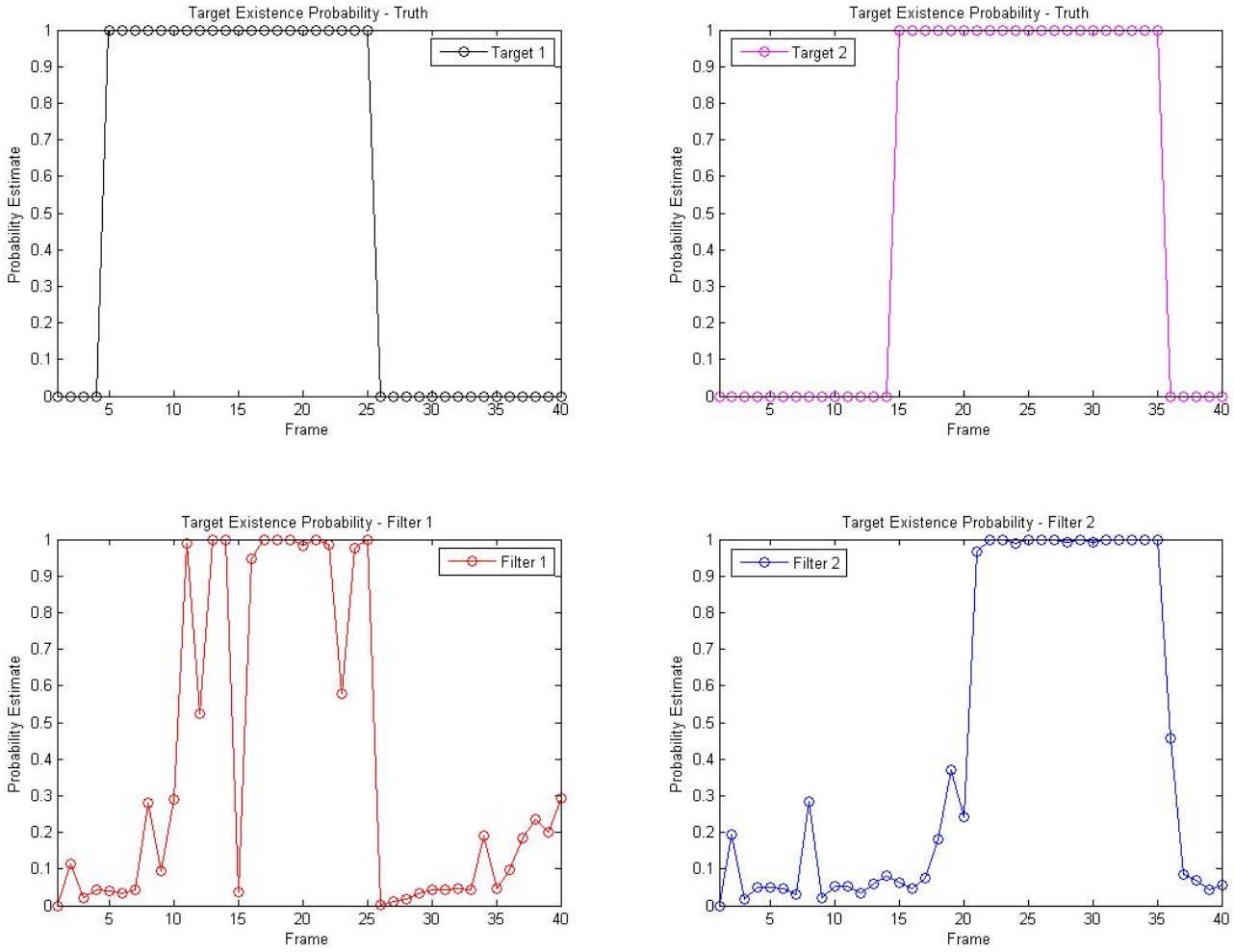

Figure 9-4: Target existence probability for two-target scenario

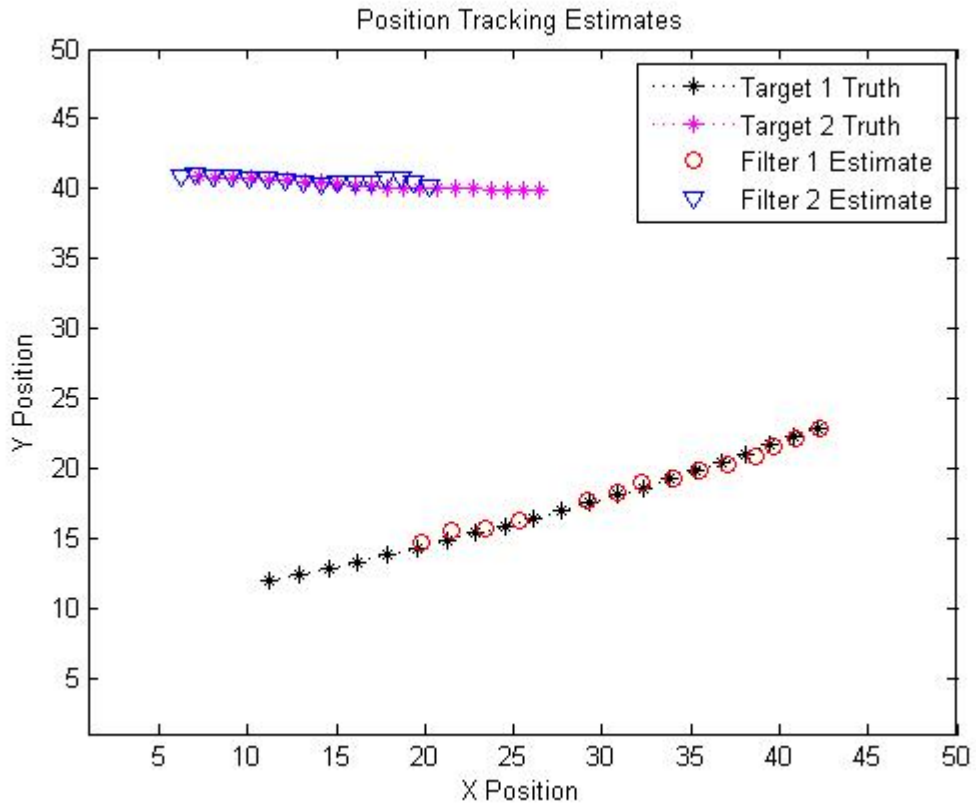

Figure 9-5: Position tracking estimates for two-target scenario

The target existence probability for the scenario is shown in Figure 9-4. Figure 9-5 shows the position estimates of each filter when the target existence probability is above 0.4. Using this threshold value, the algorithm performs quite well in detecting and tracking the two targets.

10. Conclusions and Future Work

10.1. Particle Filters

Particle and Kalman filters were compared in two types of tracking problems, linear and nonlinear. In the linear case, increasing the particle set size by a factor of 50 improved the particle filter performance by 10%. However, the Kalman filter provided superior performance. This is expected as the Kalman filter is the optimal solution for a linear Gaussian estimation problem. In the nonlinear case, increasing the particle set size by a factor of 4 improved performance of the particle filter by 28%. The particle filter outperformed the extended Kalman filter by a significant margin. The particle filter algorithm provides a robust method to handle difficult nonlinear tracking problems.

10.2. Tracking a Single Stealthy Target

A particle filter method, known as Track-before-Detect, for tracking a stealthy target in simulated EO imagery was developed. The performance analysis showed that this method is robust to target SNR and provides a means to track targets that are not detectable by image thresholding. Targets at or above a SNR of 12 dB can be reliably detected and tracked with this method. The RMS position tracking error approached 5 pixels. Using a data-driven particle proposal method, the detection results were improved and the target was detected earlier.

10.3. Tracking Multiple Stealthy Targets

The Track-before-Detect method was extended to track multiple stealthy targets. The method successfully tracked two independent targets, providing detection and position tracking estimates. The detection results show that the filter did not detect the targets until 5 frames after they appeared. Once detected, the filter provided close estimates of the true target position.

10.4. Future Work

The Track-before-Detect method uses a standard SIR particle filter for target state estimation. There are many forms of the particle filter that may provide better results than the method explored in this thesis. The method for tracking multiple targets outlined here provided good tracking results, but there are certainly other ways to approach the problem. This area has been the focus of much research and many innovative approaches exist in the literature.

Bibliography

[1] Stimson, George. *Introduction to Airborne Radar*. Raleigh, North Carolina: SciTech Publishing Inc., 1998

[2] Ristic, Branko, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Norwood, MA: Artech House, 2004.

[3] Kiefer, Jessica. "Target Tracking Using Various Filters in Synthetic Aperture Radar Data and Imagery". *Master's Thesis*. California Polytechnic University, San Luis Obispo: 2009.

[4] N. Gordon, D. Salmond, A. Smith. "Novel approach to nonlinear/non-Gaussian Bayesian state estimation". *IEE Proceedings-F*, vol. 140, no. 2, pp. 107-113. April 1993.

[5] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking." *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174-188. February 2002

[6] M. Bolic, P. Djuric, S. Hong. "Resampling Algorithms for Particle Filters: A Computational Complexity Perspective." *EURASIP Journal on Applied Signal Processing*, vol 15., pp. 2267-2277. 2004.

[7] Doucet, Arnaud, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice*. Verlag, New York: Springer 2001.

[8] Bar-Shalom, Yaakov, X. Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation*. New York, New York: John Wiley & Sons Inc., 2001.

[9] Blackman, Samuel and Robert Popoli. *Design and Analysis of Modern Tracking Systems*. Norwood, MA: Artech House, 1999.

[10] M.S. Arulampalam, B. Ristic, N. Gordon, T. Mansell. "Bearings-Only Tracking of Manoeuvring Targets Using Particle Filters." *EURASIP Journal on Applied Signal Processing*, vol. 15, pp. 2351-2365. 2004.

[11] R. Karlsson. "Various Topics on Angle-Only Tracking using Particle Filters." *Divison of Communication Systems. Linköpings universitet.* October 2002.

[12] D.J. Salmond and H. Birch. "A particle filter for track-before-detect." *Proceedings of the American Control Conference*, pp. 3755-3760. June 25-27, 2001.

[13] M. Rollason and D. Salmond. *A particle filter for track-before-detect of a target with unknown amplitude*. QinetiQ. Hants, UK.

[14] M. Rutten, B. Ristic, and N.J. Gordon. "A Comparison of Particle Filters for Recursive Track-before-detect." *7th International Conference on Information Fusion*, pp. 169-175. 2005.